AD-A044 605 DIALOG SYSTEMS INC CAMBRIDGE MA F/G 17/2
KEY WORD CLASSIFICATION.(U)
AUG 77 S L MOSHIER, P N LEIBY, R E SMITH F30602-75-C-0171
UNCLASSIFIED RADC-TR-77-122 NL

1 OF 2
ADA044605

RADC-TR-77-122
Final Technical Report
August 1977

KEY WORD CLASSIFICATION

Dialog Systems, Inc.

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

NOTICE

This document contains descriptions of inventions owned by Dialog Systems, Inc. which are protected by United States and foreign patents. No license to use or practice such inventions for sale or profit is granted hereunder.

Additional inventions disclosed herein are protected by issued or pending patents owned by or licensed to the United States Government.

This report contains a large percentage of pages which are not of the highest printing quality but because of economical consideration, it was determined in the best interest of the government that they be used in this report.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED:

ROBERT A. CURTIS, Captain, USAF
Project Engineer

APPROVED:

HOWARD DAVIS
Technical Director
Intelligence & Reconnaissance Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DAP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL
1776-1976

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-77-122	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) KEY WORD CLASSIFICATION		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) S. L. Moshier P. N. Leiby R. E. Smith		8. CONTRACT OR GRANT NUMBER(s) F30602-75-C-0171
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dialog Systems, Inc. 639 Massachusetts Avenue Cambridge MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 31011F 70550722
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRAP) Griffiss AFB NY 13441		12. REPORT DATE August 1977
		13. NUMBER OF PAGES 144
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same 148p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
Robert A. Curtis (IRAP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech Processing
Key Word
Word Spotting

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes the development of a real-time key word recognition. The basic objectives for the system were the ability to detect one or more key words in continuous speech, independent of the speaker.

By concatenating half-syllable sized utterances in sequence to detect the occurrence of a spoken word, good temporal registration was obtained and with good recognition results.

(cont'd)

392 051

On a limited number of talkers (about ten) results of 90% - 95% detection and 4-6 false alarms per hour were obtained. When the data base is increased to approximately fifty talkers, the results obtained are 85% detection and 20-25 false alarms per hour. The techniques and hardware developed under this effort appear very promising. Improvements appear feasible with only modest changes.

TABLE OF CONTENTS

EVALUATION

This report summarizes the development of a real-time keyword recognition system with the basic objective of detecting when a keyword occurs in continuous speech independent of speaker. The initial speaker independent results (over 50 different speakers) of about 85% detection and 20-25 false alarms per hour are very promising. The ability to perform reliable keyword detection would greatly enhance the ability to perform other related speech signal processing tasks, such as speaker identification and continuous word recognition.

Robert A. Curtis

ROBERT A. CURTIS, Captain, USAF
Project Engineer

SUMMARY

This report summarizes the development of a spoken key word recognition system developed by Dialog Systems, Inc. for Rome Air Development Center under contract F30602-75-C-0171. The basic objectives for the system were the ability to detect one or more key words in continuous speech, independent of language, speaker, or spoken text. The development was to be initially performed in the English language and with the effectiveness goal of 90% detection of key words and 5 or less false alarms per hour. The system was also required to work over telephones lines and radio links without being susceptible to spectral equalization or distortion. The resulting device could then be used to monitor all types of broadcast material and by a proper selection of key words, to determine the gist of the real-time or recorded speech.

Key word spotting differs from continuous speech understanding in that one trains a word spotting machine to understand one or a few words and hopes that it will reject all other words without actually having to understand any of them. In passing from simple closed-vocabulary recognition to word spotting, one encounters the following major problems:

1) The relative number of wrong choices (potential false alarms) that must be rejected is very large since the range of words and phrases that might be put to the machine is unlimited.

2) There is no known reliable method of segmenting the continuous speech material into words or syllables on the basis of short term acoustic cues.

3) The acoustic description of a word changes with the verbal context in which it appears. The relative timing and duration of events can vary radically; the phonetic character of sounds at the beginning and end of the word can be modified strongly by the preceding and following words; and whole syllables are sometimes omitted or other sounds added.

Dialog's starting point in this program was a previously-developed algorithm which recognizes single words spoken in isolation with high talker-independent accuracy over unknown telephone lines. This algorithm was extended to recognize continuous speech by detecting half-syllable sized utterances in sequence. A high-speed digital vector arithmetic processor was designed and constructed to perform the lengthy calculations required in real time, and the software was re-written for the new machine.

After the vector processor was fabricated and programmed, the key word recognition effort concentrated on a 6-minute script consisting of a hypothetical news broadcast, which included the word "Kissinger" in four different context, a set of airline-to-ground exchanges, and a short list of random numbers. About 77 renditions of the script were obtained in English, 3 in Spanish, 2 in French, and one in Chinese, all over the telephone. At about the time a data base was being made up of these voices, Dialog received 13 renditions of the script on wide-band tape recordings. These did not appear to be compatible with the telephone voices so the main effort was switched to these tapes. A data base was then made of the four "Kissinger" renditions from each of nine speakers and this material became the subject of an intense development effort. Four voices were held out of the data base as test inputs.

The resulting word spotting technique was capable of operating in real time for at least two key words simultaneously, and obtained an effectiveness of 90%-95% detection of a single key word, with 4-6 false alarms per hour against the limited number of test voices. It was found that the limited data base could not hold this accuracy against a wider population of test voices; therefore the data base was increased to 41 voices from additional wide band tape recordings. Ten test voices, not included in

the data base, were used for a series of runs under various test conditions.

Although some parts of the original isolated word algorithm have yet to be included in the word spotting tests, the results to date are good for a moderately wide population of talkers. As shown in Table A the system could be made to operate in the untrained mode at 83% detection of key words and 24 false alarms per hour or 70% detection and 6 false alarms per hour depending on the threshold settings for each pattern. By training the machine on the first rendition of the key word the characteristics improved somewhat.

Although this performance is only marginally adequate for operational equipment, the general ca ability of this technique in key word detection appears very promising towards meeting its objectives with further development.

The variability of target word character and of material to be rejected is quite large, and predictions based on experience with as many as 41 different talkers are subject to large statistical errors. In the course of this project, statistical models have been developed for several aspects of the recognition process, These models, discussed in a latter section, seem to shed some light on the problems of implementing and testing improved speech processing algorithms.

	98% DETECTION PER PATTERN				95% DETECTION PER PATTERN				93% DETECTION PER PATTERN			
	Untrained		Trained		Untrained		Trained		Untrained		Trained	
Talker	DET	FA	DET	FA	DET	FA	DET	FA	DET	FA	DET	FA
KN	4	6	3	9	4	1	2	2	1	0	3	0
RJL	3	2	2	2	3	1	2	0	3	2	2	0
GDP	4	2	2	1	2	0	2	0	2	1	2	0
FGH	2	1	3	0	2	0	3	0	2	0	3	1
EM	4	5	3	0	4	1	3	0	3	0	3	0
JN	4	0	3	0	3	0	3	0	4	0	3	0
AG	0	3	2	1	0	2	1	0	0	0	1	0
WH	4	1	2	0	3	0	2	0	3	0	0	0
JMT	4	1	3	0	4	1	2	0	3	0	2	1
HK	4	3	3	0	3	0	2	0	2	0	2	0
Totals	.83	24	.87	13	.70	6	.73	2	.58	3	.70	2

Table A. Detection (DET) and false alarm (FA) data for the real-time word spotting system. Under each condition the number of possible detections per talker is 4 in the untrained case and 3 in the trained case. Total elapsed time for all subjects is 1 hour.

I Experimental Study And Analysis

1. Summary of the Word Spotting Algorithm

A block diagram of the major processing steps of the continuous speech recognition procedure is shown in Figure 1. The analog speech waveform is digitized with 12-bit resolution at an 8 KHz sampling rate. Spectrum frames having 32 points (129 Hz spacing between sample frequencies) are computed every 10 milliseconds in block A of Figure 1. The spectrum analysis involves taking the cosine transform of the output of a hardware autocorrelator at each desired frequency. The correlator is provided so that alternate predictive coding implementations can be used. Smoothing in frequency and time also occurs at A, primarily to remove abrupt transitions in the spectrum and aliasing with the pitch period of the voiced portions of the speech.

In the next processing steps the sequence of spectrum frames is normalized and enhanced with respect to frequency, amplitude and time. The first step is to accumulate a gross picture of the input spectrum, at B, which is then used to adapt the system to the unknown frequency response of the communications channel. Provision for subtracting background noise can be made at this point, but is not yet implemented in a real-time system. Since the channel equalizer uses received speech to estimate the channel response, some of

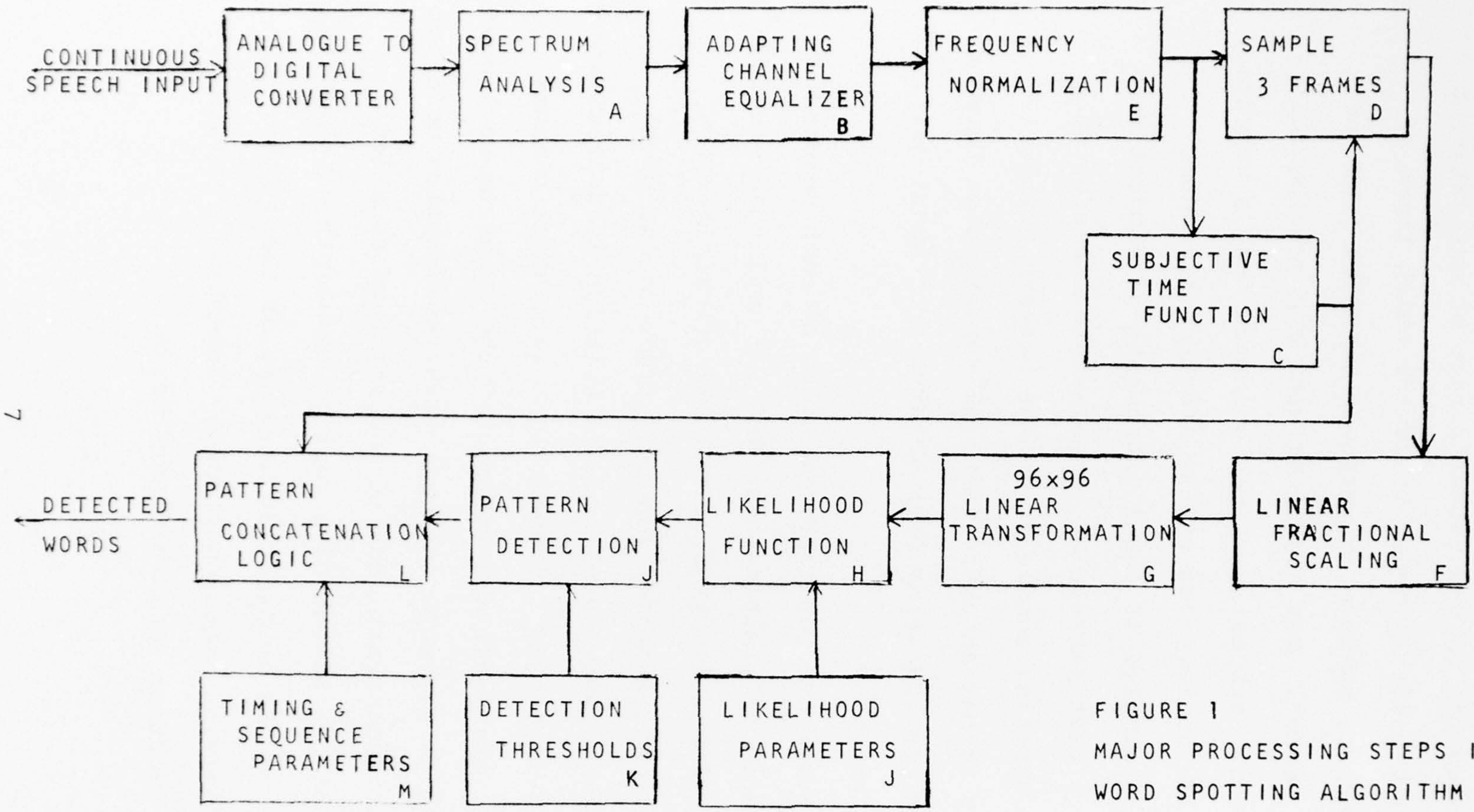


FIGURE 1
MAJOR PROCESSING STEPS IN THE WORD SPOTTING ALGORITHM

the between-talker variation is removed by the equalizer. Bias and fluctuation in the talker's rate of articulation must be accounted for in order to decide which frame of a stored reference pattern corresponds to which frame of the incoming speech. Our algorithm uses a psychologically-based measure of "subjective time" computed from a weighted sum of the time derivatives of the spectrum elements. The frames are sampled at equal increments of subjective time to form a test pattern of three samples; every input frame is the starting sample of a test pattern. This method yields rather accurate temporal registration between reference and test patterns over intervals of about half a spoken syllable.

In block F the position and width of each spectrum line is enhanced by limiting high and low amplitude spectrum coefficients with a function of the form (1-x)/(1+x). The center of this transfer function rides up and down with the average power level and the function is approximately logarithmic over a range of about 10 to 1. After channel equalization, the peaks and valleys of the spectrum no longer correspond to an all-pole rational transfer function model of the talker's vocal tract, and it appears that detailed measurement of the height of a peak or the depth of a valley adds nothing to the recognition accuracy of the system. The chosen limiting function strongly resembles a typical firing rate function of an auditory nerve.

Transformation of the frequency axis, shown at E, is not yet implemented. Some form of frequency rescaling may be expected to improve the speaker-independent recognition accuracy if, for example, a method of adapting the transformation to the voice quality without explicit training can be found.

Up to this point the algorithm has paid no particular attention to the content of the input signals. Now, however, having normalized and enhanced the data with respect to its important physical dimensions, the system makes an effort to enhance the phonetic content of the spectrum patterns. This is implemented by linearly projecting the data from the frequency domain into an abstract space in which sounds belonging to different phonetic speech classes are maximally separated. The 32 spectrum coefficients from the three sampled frames comprise a 96-element vector which is transformed by matrix multiplication into the new space. The coefficients of the transformation matrix are constants evaluated by factor analysis of labeled training data. In our isolated word recognition algorithm this type of transformation is applied to each 32-element sample frame and is found to improve overall accuracy by a factor of two or more. It has not yet been implemented in the continuous speech algorithm. An important potential benefit of processing several successive frames in this manner is that the cross-correlations of all frames at all pairs of frequencies are taken into account.

The input pattern is now considered ready to be matched against a set of reference templates, using the statistical behavior of the reference data to determine the closeness of fit to each template. From a study of a large number of samples we have concluded that the patterns at this processing stage are adequately modeled by Gaussian distribution functions. To detect the occurrence of previously learned patterns we implement a likelihood reciever based on Gaussian statistics. Decision thresholds for the resultant likelihood functions are calculated from the statistics of measured likelihood scores of labeled target patterns.

This completes the recognition procedure for intervals of half a syllable of speech. To recognize a sequence of patterns in the word spotting task we set the decision thresholds for very high detection probabilities and depend on subsequently applied concatenation rules to reject false alarms. The sequence of detected patterns must match a list of permissible "spellings" of the target word, and in addition each detection must happen within prescribed real and subjective time bounds relative to other pattern detections. This section of the algorithm is under active development, and is expected to change significantly as more experimental work is completed.

A detailed flow chart of the algorithm is presented in Chapter II.

2. Statistics of the feature measurements

From the point of view of a given pattern recognition algorithm, departures of the measured input features from perfect pattern matches arise from unknown and unpredictable sources and are to be treated as random variables. If this were not the case, one could show that closer pattern matches are possible by implementing an improved algorithm prior to the pattern matcher. It is therefore important to analyze the sample distribution functions of the measurements in order to find the best probabilistic decision procedures to use and to help discover new deterministic factors that can improve the recognition algorithm.

The accompanying figures show frequency of occurrence histograms for individual feature measurements (single coordinates of the output of box F or box G, Figure 1), for repeated applications of a particular speech sound embedded in the speech of many different talkers. The typical histogram, Figure 2, has a nondescript bell curve shape. A χ^2 test applied to the top curve of Figure 2, for example, indicates that the probability of getting a random departure this large from a fitted normal distrubution is 0.7 (the total number of samples is 60).

To get a more precise estimate of the average shape

Figure 2. Typical distributions of speech parameter data. The horizontal axis is scaled in such a way that the mean value is centered in the display and the range shown is plus and minus 4 standard deviations.

of the distributions a composite sample frequency histogram was developed by summing a large number of the histograms like Figure 2 after scaling the horizontal axis of each by subtracting its mean value and dividing by its standard deviation. Thus if all the distributions were rectangular, the composite would be rectangular, etc. The resulting frequency function, Figure 3, representing some 230,000 events, makes a very close fit to a Gaussian curve and suggests that the likelihood processor should employ a Gaussian model for the distributions of pattern data.

It is not immediately clear why the the sample frequency functions have a Gaussian shape. The signal processing itself contains some averaging which by the central limit theorem would tend to produce normal distributions. It will require further study to determine whether or not the observed functions are actually produced as an artefact of the measurement process. In any event some of the sample distributions have had a decidedly non-Gaussian form, and these unusual distributions will be discussed in more detail.

Figure 4 shows a distribution having a tall peak with some broadly distributed data to the left of it. This histogram represents the distribution of scaled spectral power (output of box F, Figure 1) at 4 KHz for the initial /n/ sound in the word /nine/. Because 4KHz is

Figure 3. Composite statistical frequency function made by summing 3,840 of the curves illustrated in Figure 2 of the distributions of various measurements within phonetically specific classes.

On the horizontal axis the class intervals are spaced at intervals of 1/32 standard deviation; in the other figures the intervals are 1/4 standard deviation. Otherwise, the horizontal scaling is the same.

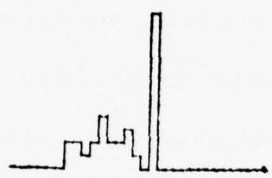


Figure 4. Unusual distribution arising from a systematic processing error.

above the upper cutoff frequency of the signal aliasing filter and the /n/ is a relatively low-amplitude sound, the computed spectrum was noisy except at the format frequency peaks due to truncation errors in the analogue to digital converter. The curve seems to be a composite of two distributions --first, the tall spike produced by those samples having sufficient energy to yield an accurate spectrum, and second, the relatively much broader distribution of low energy noisy samples caused by the logarithmic scaling tending to a large negative number as the amplitude goes to zero. This is a clear case of a processing artefact, and was actually not noticed until these histograms were produced and examined. On installing a higher resolution analogue to digital converter distributions of this kind no longer occured.

A second type of strange distribution is illustrated by several examples in Figure 5. The chance of getting such large departures at random from a normally distributed population ranges from about 10^{-4} to less than 10^{-7}. These curves show two distinct peaks, which turn out to be related to two distinctly different pronunciations of the target sounds. The bimodality is eliminated by partitioning the sample space into two new classes and making a separate reference pattern for each distinct pronunciation.

Figure 6 illustrates another processing artefact,

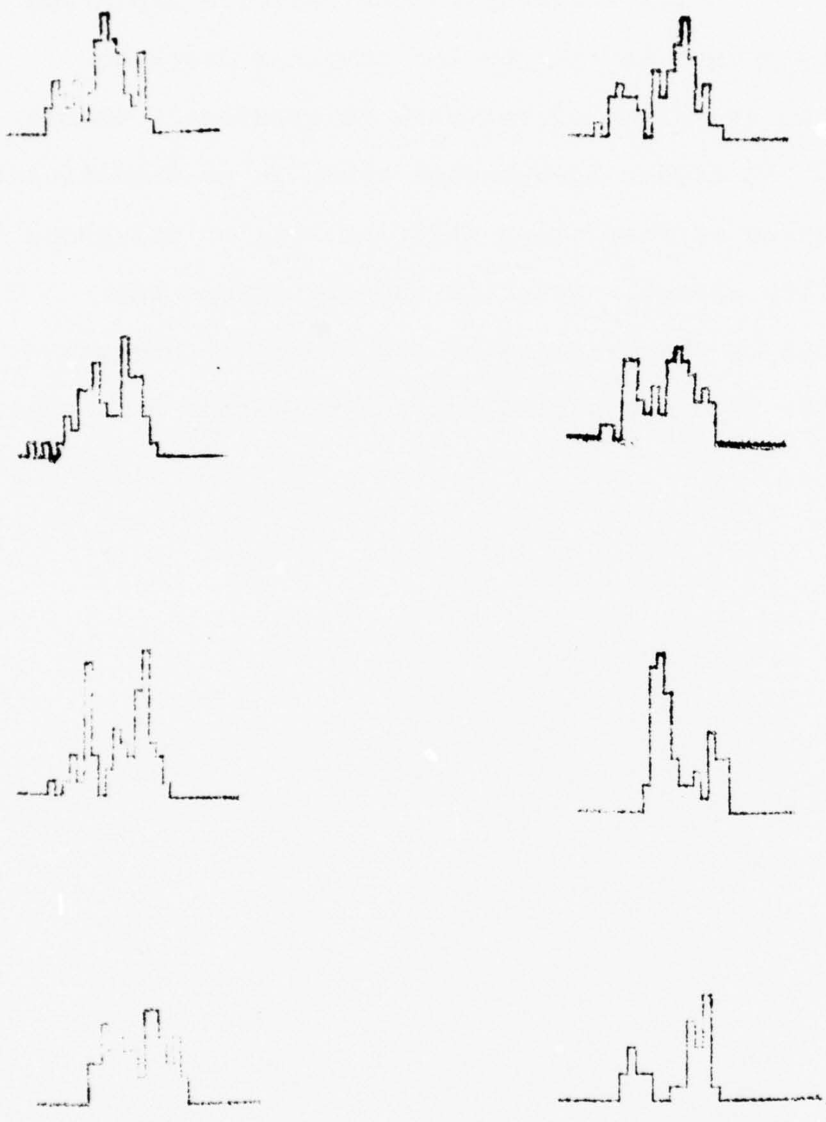


Figure 5. (clockwise from lower left). Examples of bi-modal frequency histograms.

a difference between processed telephone speech and high fidelity speech. At low frequencies the relative amplitude of high quality speech as seen by the computer drops so rapidly that the equalization software is apparently unable to compensate. At higher frequencies there is no significant difference between corresponding distributions of telephone and high fidelity speech. Actually the difference was brought about by the presence of low frequency hum and noise in all the recordings used for the telephone data base.

Figure 6. Distribution of observed spectral amplitude at constant frequency for a fixed utterance over the data base of high quality speech. The histograms are scaled so that the mean amplitude for telephone speech is centered horizontally in each display, and the horizontal width of each display is plus and minus 4 standard deviations referred to telephone speech.

Left: Typical of low frequencies (0 and 150 Hz. spectrum terms).

Right: Typical of higher frequencies.

3. Experimental data and probability models

Detection probability

For one pattern $\underline{x} = (x_1, x_2, \ldots, x_{96})$ the likelihood function relative to the kth reference template is

$$\lambda_k(\underline{x}) = \sum_{i=1}^{96} \ln \sigma_{ik} + \frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2},$$

where μ_{ik} and σ_{ik} are the mean and standard deviation, respectively, of x_i given that $\underline{x}$ is a sample of the kth speech sound class. The model is Gaussian and assumes uncorrelated variables, so λ ideally is the sum of 96 independent normal deviates and has a χ^2 distribution. The situation is complicated by the fact that we use sample estimates of the means and variances; this produces an "offset χ^2" function. In addition the x_i are of course not quite independent. However because the number of degrees of freedom is large $\lambda_k(\underline{x})$ should have an approximately normal distribution. Figure 7 shows sample detection functions for the training sets of several templates, plotted on a probability scale in which a normal distribution would be represented by a straight line with slope 1.

Figure 8 illustrates a case in which the joint detection and false alarm probabilities for two speech patterns

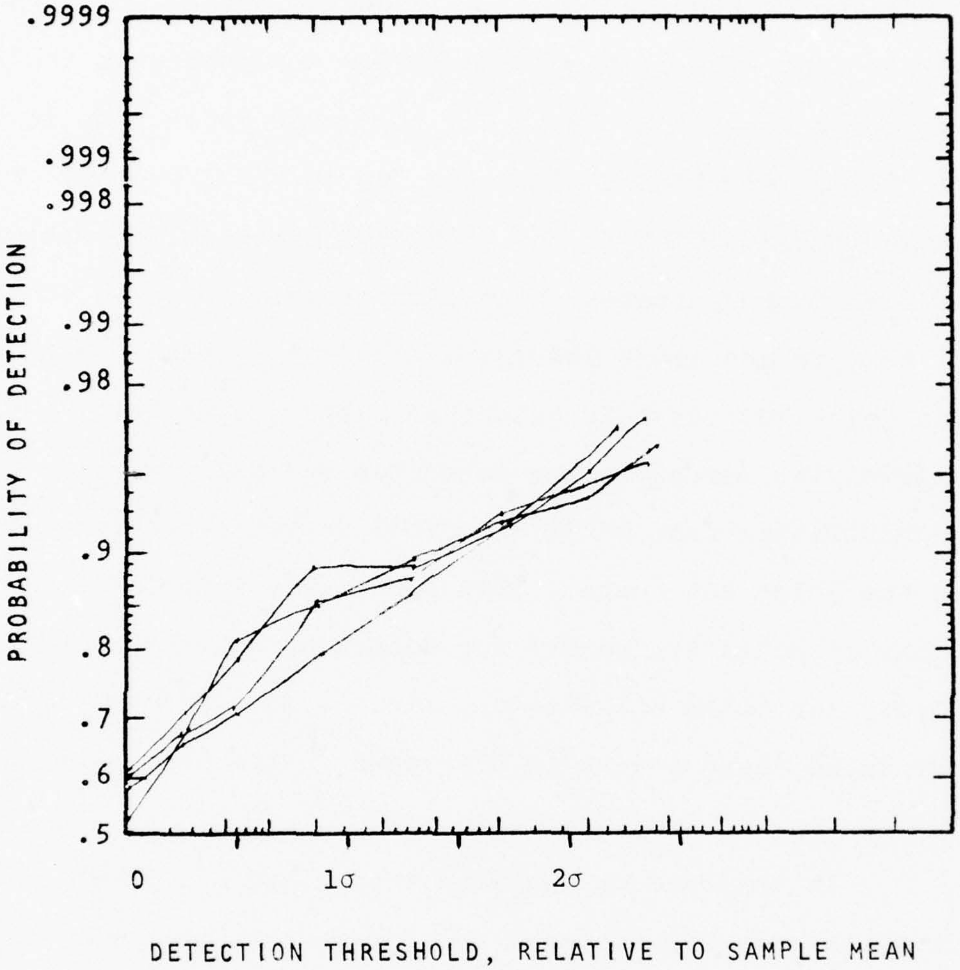


Figure 7.

Detection curves for several 96-element speech patterns.

behave as if the events were independent. The curve labeled SYL31 represents a sequence of spectrum frames beginning in the initial /th/ of the word /three/ and ending in the /r/. The curve labeled SYL32 is for the sound which ends in the /i/ of /three/ and begins after the /r/ in an /I/ sound. The curve labeled JOINT is the joint ROC curve for detection of both sound patterns in correct temporal order. There are 31 targets and about 290 false alarms possible in each case. The joint ROC curve is predicted rather accurately by simply multiplying corresponding detection and false alarm probabilities from SYL31 and SYL32 to get a predicted point on the joint ROC curve. This apparently independent behavior holds frequently for detection statistics but rarely for false alarm statistics. A more general model for false alarm events is discussed in the next section.

In the case of a closed vocabulary set in which templates for all received sounds are available the situation is somewhat different. This case is worth discussing because as we increase the number of reference patterns we attain an increasingly fine covering of the space of all speech sounds. Ordinarily the likelihood $\lambda_k(\underline{x}|\underline{x}$ is in class $k)$ is positively correlated with the other likelihood functions $\lambda_j(\underline{x})$ of the same argument. Nevertheless, experience with closed vocabularies indicates that if

$$\lambda_m(\underline{x}) = \min_{j \neq k}\{\lambda_j(\underline{x})\},\ \underline{x} \text{ in class } k$$

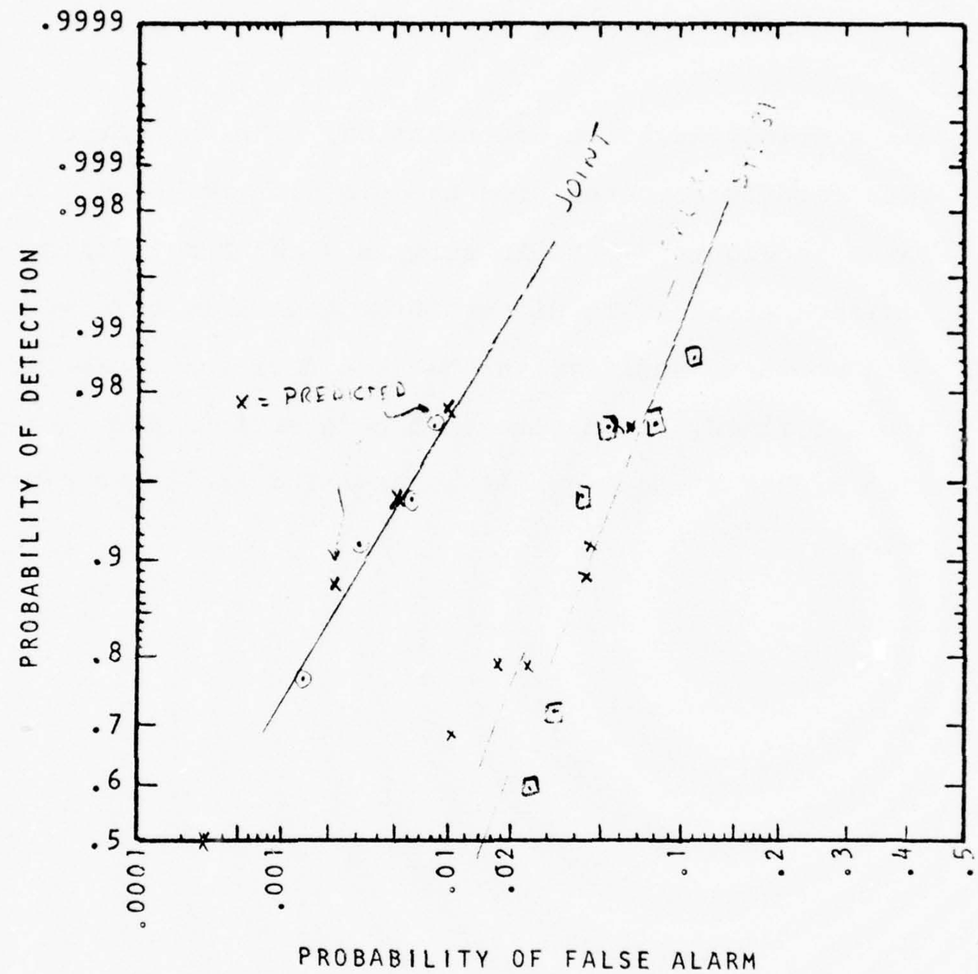


Figure 8. ROC curve for detection of the word /three/ in an environment of isolated digits using the continuous speech algorithm. The curves are derived from 32 male voices recorded over a variety of standard telephone connections.

then

$$\Delta_k(\underline{x}) = \lambda_k(\underline{x})-\lambda_m(\underline{x})$$

has a nearly Gaussian distribution. The ROC curve is then readily computed from the distribution of Δ, as shown in Figure 9. While this is a maximum likelihood decision strategy it differs from the currently implemented word spotting strategy in that the decision thresholds are not fixed. This permits the detection rate to be much higher without greatly increasing the false alarm rate.

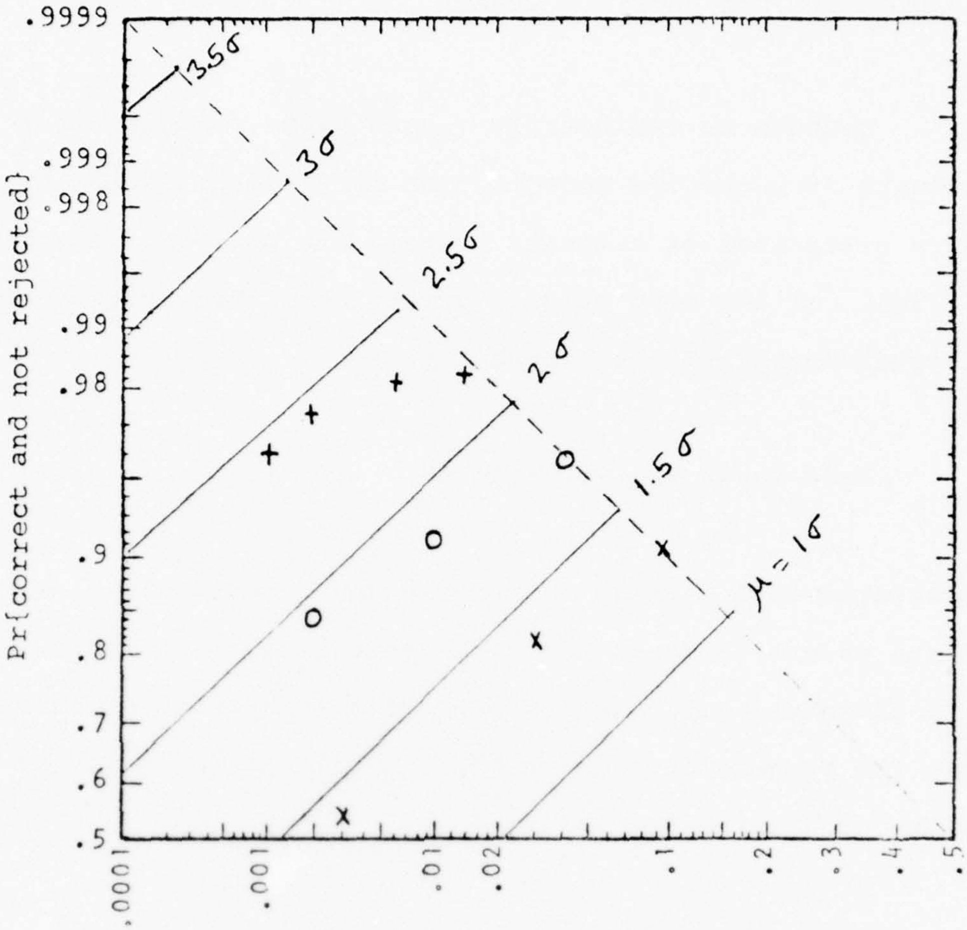
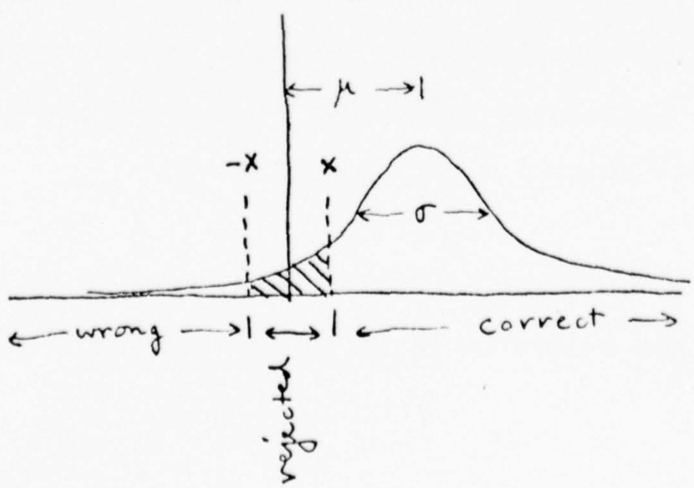


Figure 9. ROC curves for a quasi forced-choice decision rule. Experimental data are plotted for a selected 8-word vocabulary (+), the 10 digits (o), and a 34-word vocabulary (x) in a discrete word recognition task with telephone speech and many talkers.



Probabilistic false alarm model

Because an arbitrarily chosen pattern might occur anywhere in connected conversation it will be assumed that false detections of a single pattern are uniformly distributed in time. If the time axis is partitioned into short intervals of equal duration, the probability

$$\Pr\{\text{A is in the } k\text{th interval}\} = P_A$$

associated with pattern A is the same for all values of k. If the events {A is in the k_1th interval}, {A is in the k_2th interval}, ... are independent for all sets of intervals, then the probability that pattern A will not be detected in any of T intervals is

$$\Pr\{\text{A is not in interval } k_1 \text{ or } k_2 \text{ or } \ldots \text{ or } k_T\}$$
$$= (1-P_A)^T = e^{-\lambda_A T},$$

where $\lambda_A \equiv -\ln(1-P_A)$. The expected number of detections in T intervals is $P_A T$, and if P_A is small then P_A approaches λ_A and the distribution of the number of detections in T intervals becomes Poisson with mean value $\lambda_A T$.

The joint detection of two patterns A and B is a coincidence of the events {A is in the kth interval} $\equiv$ {A}

and {B is in the jth interval} $\equiv$ {B}. If the events were independent their joint detection probability would be

$$\Pr\{A \text{ and } B\} = P_A P_B,$$

and the expected number of coincidences in T time intervals would be $P_A P_B T$. Experiment shows, however, that this estimate is too small, typically by a factor of 2 or 3. To arrive at a closer estimate we must admit that the events are not independent and consider the conditional probability

$$\Pr\{B|A\} = \frac{\Pr\{A \text{ and } B\}}{\Pr\{A\}}.$$

The experimental result noted above suggests that we try to put

$$\Pr\{B|A\} = \alpha \Pr\{B\}, \tag{1}$$

with the parameter α between 2 and 3. This formula asserts that if pattern A has been detected, then pattern B is α times more likely to be detected in the specified coincident time interval than in a randomly chosen interval.

To extend the model to cover multiple events without introducing additional parameters, we will suppose that the conditional probability of the nth event in a sequence depends on

the immediately preceding event sought, but not on any of the earlier events. The first event is uniformly and independently distributed in time as before. Thus

$$\begin{aligned}&\Pr\{A_n \mid A_1 \text{ and } A_2 \text{ and } \dots \text{ and } A_{n-1}\}\\ &= \Pr\{A_n \mid A_{n-1}\} = \alpha \Pr\{A_n\}. \end{aligned} \qquad (2)$$

Under these assumptions the probability P_n of jointly detecting n specified patterns in n associated coincidence intervals is

$$\begin{aligned}P_n &= \Pr\{A_1 \text{ and } A_2 \text{ and } \dots \text{ and } A_n\}\\ &= \alpha^{n-1} \Pr\{A_1\} \Pr\{A_2\} \cdots \Pr\{A_n\}, \end{aligned} \qquad (3)$$

and the expected number of joint detections in T intervals associated with A_1 is $P_n T$. Thus our model is of joint false alarms as Markov chains.

Since the unconditional probabilities $\Pr\{A_i\}$ vary from pattern to pattern, the predicted results for joint detections of several patterns do not lie on a simple curve. Table I, however, permits a numerical comparison of the theoretical model with experiment. The unconditional probabilities for four speech patterns are tabulated on the left; these patterns are extracted from the beginning and end of the words /one/ and /six/ in contexts of continuous

Observed unconditional probability $Pr\{A_i\}$ of false detection in an interval $\approx$ 0.17 second	Event:	Joint false alarm rate in 4.44 minutes of speech, predicted by equation (3) with α=3.0:	Observed false alarm rate:
$Pr\{A_1\} = 0.051$	$\{A_1,A_2\}$	25.3	29
$Pr\{A_2\} = 0.110$	$\{A_1,A_2,A_3\}$	6.3	3
$Pr\{A_3\} = 0.084$	$\{A_1,A_2,A_3,A_4\}$	3.9	3
$Pr\{A_4\} = 0.205$	$\{A_1,A_2,A_3,A_4,A_3,A_4\}$	0.6	0

Table I. Comparison of observed false alarm rates for multiple pattern phrases in continuous telephone speech with the predictions of equation (3).

digits. Detection thresholds and coincidence intervals were adjusted for >90% correct detection of the multiple event $\{A_1, A_2, A_3, A_4\}$ which represents the phrase /one six/ (part of a spoken telephone number). The false alarm data were gathered from a set of two-second segments taken at random from recordings of 15 male subjects reading a script over various switched telephone network connections. The false alarm data did not contain any spoken numbers; but the numbers for correct detection were taken from different portions of the same recorded scripts. Four increasingly long joint events were set up in the form of computer parameters and the false alarm data base of 4.4 minutes' cumulative duration was searched for possible false alarms. It can be seen from the table that the predicted and observed numbers of false alarms compare favorably.

Equation (3) has a serious flaw, because if the individual event probabilities are large the predicted joint probability can be greater than 1. This trouble can be handled readily by assuming that the conditional events $\{B|A\}$, like $\{A\}$, obey Poisson statistics. Then if we put

$$\lambda_{B|A} = \alpha \lambda_B,$$

we are asserting that the Poisson rate function λ increases by the factor α whenever pattern A occurs.

Under this assumption equation (3) becomes

$$P_n = \Pr\{A_1\} \prod_{i=2}^{n} (1 - e^{-\alpha\lambda_i}) \tag{4}$$

where λ_i is the rate function associated with the event A_i. For comparison purposes equation (4) was used to generate predicted false alarm rates given in Table II, which relates to an experiment similar to the one described above but with generally more diffuse likelihood functions.

Event:	Joint false alarm rate predicted by equation (3) with $\alpha=2.4$:	Joint false alarm rate predicted by equation (4) with $\alpha=2.4$:	Observed joint false alarm rate:
$\{A_1, A_2\}$	27.5	24.7	27
$\{A_3, A_4\}$	55.3	80.8	67
$\{A_1, A_2, A_3\}$	4.6	3.5	4
$\{A_1, A_2, A_3, A_4\}$	2.6	1.5	4

Table II. Comparisons of observed false alarm rates with predictions made by equations (3) and (4). This experiment differed from the one reported in Table I in that the spacing between speech sample frames was reduced from 20 (arbitrary) units of subjective time to 12 units. The closer spacing tended to increase the effective dispersion of the raw measurements, with the result that the false detection probabilities for single pattern events increased.

Variance of estimated detection probability

Each of N talkers makes n trials of a target word. We assume that the ith talker has a certain detection probability p_i for the target word, and that the trials are independent. Then the number x_i of detections has a binomial distribution with mean value

$$E\{x_i | p_i\} = np_i \tag{1}$$

and variance

$$\sigma^2_{x_i} = np_i(1-p_i) . \tag{2}$$

From the total set of nN trials we estimate the detection probability for the whole population of talkers, $\hat{p}$, from

$$x \equiv \sum_{i=1}^{N} x_i , \quad \hat{p} = \frac{x}{nN} , \tag{3}$$

and we want to know the variance of the estimated probability $\hat{p}$:

$$\sigma^2_{\hat{p}} = E\{\hat{p}^2\} - E\{\hat{p}\}^2 = \frac{1}{n^2N^2} \left(E\{x^2\}-E\{x\}^2\right) . \tag{4}$$

The expectations in (4) are $E\{x\}=NE\{x_i\}$, $E\{x^2\}=N^2E\{x_i^2\}$ -- the latter because the x_i are uncorrelated. We find $E\{x_i\}$ by integrating the conditional expectation as follows:

$$E\{x_i\} = \int E\{x_i|p\} f_p(p)\, dp \tag{5}$$

where $f_p(p)$ is the probability density function of talkers' detection probabilities -- i.e., the relative probability of finding a talker whose detection rate is p.

From (1) and (5),

$$E\{x_i\} = \int n p_i \; f_{p_i}(p_i) \; dp_i = nE\{p_i\} \tag{6}$$

which gives one of the terms needed in (4). To get the other term we first compute

$$\begin{aligned} E\{x_i^2|p_i\} &= \sigma^2_{x_i} + E\{x_i|p_i\}^2 \\ &= np_i(1-p_i) + n^2\mathbf{p}_i^2 \\ &= n(n-1)p_i^2 + np_i. \end{aligned} \tag{7}$$

Then by equation (5),

$$\begin{aligned} E\{x_i^2\} &= \int n(n-1)p_i^2 \; f_{p_i}(p_i) \; dp_i \;\; + \int np_i \; f_{p_i}(p_i) \; dp_i \\ &= n(n-1)E\{p_i^2\} + nE\{p_i\}. \end{aligned} \tag{8}$$

Substituting into (4) and collecting terms yields the result

$$\sigma^2_{\hat{p}} = \frac{n-1}{nN}\, \sigma^2_{p_i} + \frac{1}{nN}\, E\{p_i\}(1-E\{p_i\}). \tag{9}$$

The first term in (9) is the contribution of the population variance on the assumption that the detection probability of each talker is accurately known; for large n, equation (9) approaches the result that would be obtained by direct application of the central limit theorem. On the other hand, if $n=1$ the result has the same form as the variance of the binomially distributed $\hat{p}$. From (2), the variance of $\frac{X}{N}$ given that $n=1$ and $p_i=E\{p_i\}$ is equal to

$$\sigma^2_{\hat{p}|n=1} = \frac{1}{N} E\{p_i\}(1-E\{p_i\}). \tag{10}$$

Thus

$$\sigma^2_{\hat{p}} = \frac{n-1}{nN} \sigma^2_{p_i} + \frac{1}{n} \sigma^2_{\hat{p}|n=1}. \tag{11}$$

If the total number nN of trials is fixed, equation (9) shows that the best experimental result should be obtained by taking one trial from each talker so as to maximize the number of talkers. If the number of talkers is fixed then the number of trials per talker should be as large as possible.

Since $0 \leq p_i \leq 1$, we can get a worst case bound on both $\sigma^2_{p_i}$ and the distribution of $\hat{p}$. The worst case is the one in which the density function of p_i is concentrated at $p_i=1$ with probability $p = E\{p_i\}$ and at $p_i=0$ with probability

1-p. Then $\sigma^2_{p_i} = p(1-p)$. Setting this into equation (9) as an upper bound yields

$$\sigma^2_{\hat{p}} \leq \frac{1}{N} p(1-p) \quad \text{always.} \tag{11}$$

In the event that the p_i actually assume the worst case distribution, the distribution of $N\hat{p}$ is binomial with mean Np and variance $Np(1-p)$, reflecting the fact that if p_i is zero or one there is nothing to be learned by taking more than one sample per subject. A more optimistic case is considered in the section on confidence limits.

The sample variance

$$s^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i}{n} - \hat{p} \right)^2$$

computed from the observed values x_i is an unbiased estimator of $N\sigma^2_{\hat{p}}/(N-1)$. If the original population is normally distributed, then $t = \hat{p}(N-1)^{\frac{1}{2}}/s$ is a random variable having a Student's t distribution with N-1 degrees of freedom. This statistic can be used to find confidence intervals for $\hat{p}$ based on observations from one experiment.

Confidence limits on the sample statistics

A lower c-percent confidence limit on, for example, the detection probability is found by computing an assumed value $p_c(\hat{p})$ for $E\{p_i\}$ which is so low that the observed value of $\hat{p}$ lies in the c-th percentile of the distribution function of $\hat{p}$. Then we can bet that on repeated runs of the experiment the interval $[p_c(\hat{p}),1]$ will contain $E\{p_i\}$ c percent of the time.

To make this calculation it is necessary to assume a distribution function for $\hat{p}$. In the worst case, described by equation (11), it is binomial with number of trials N and probability $p_c(\hat{p})$. Note that this case applies exactly if $n = 1$ trial per subject. A table of binomial confidence limits is included in the Appendix to this report.

A less pessimistic estimate based on some experience results from supposing that the true population density of detection probabilities p_i is J-shaped, roughly exponential with the most likely p_i near 1 and a long tail extending to low probabilities. For such a function the variance of p_i is approximately $\sigma^2_{p_i} = (1-p)^2$. Invoking the central limit theorem we assume that $\hat{p}$ is normally distributed. For a particular value of c we will want $\hat{p}$ to be some number t of standard deviations above the mean value p_c. Evaluating

the standard deviation by substituting $(1-p)^2$ into equation (9) we find that p_c satisfies

$$\hat{p} = p_c + \frac{t[(n-1)(1-p_c) + p_c(1-p_c)]^{\frac{1}{2}}}{n^{\frac{1}{2}}N^{\frac{1}{2}}}.$$

Overall Performance Of The Algorithm

Forty-one male subjects made recordings of a six-minute script which were processed to generate seven syllable-like spectral reference patterns for the target word /Kissinger/. Each pattern had an alternate, to accommodate variant pronunciations. Any sequence of pattern detections satisfying the temporal windowing rules was counted as a detection of the target word. Detection of either alternate was permitted, regardless of which alternates for other syllables were detected. Recordings of ten additional male subjects were played into the system to test the detection and false alarm rates at various pattern detection threshold settings. All recordings were made with high-quality microphones in a relatively quiet environment. This material was used for a benchmark demonstration witnessed by RADC personnel at the end of the contract period.

Much of the development had been carried out using a subset of 9 subjects for training and 4 subjects for test data bases; scores of 90%-95% detection at 4 to 6 false alarms per hour were typical for this smaller data base. With the larger number of training and test subjects, the average scores were considerably lower, as seen in Table III and Figure 10.

	98% DETECTION PER PATTERN				95% DETECTION PER PATTERN				93% DETECTION PER PATTERN			
	Untrained		Trained		Untrained		Trained		Untrained		Trained	
Talker	DET	FA	DET	FA	DET	FA	DET	FA	DET	FA	DET	FA
KN	4	6	3	9	4	1	2	2	1	0	3	0
RJL	3	2	2	2	3	1	2	0	3	2	2	0
GDP	4	2	2	1	2	0	2	0	2	1	2	0
FGH	2	1	3	0	2	0	3	0	2	0	3	1
EM	4	5	3	0	4	1	3	0	3	0	3	0
JN	4	0	3	0	3	0	3	0	4	0	3	0
AG	0	3	2	1	0	2	1	0	0	0	1	0
WH	4	1	2	0	3	0	2	0	3	0	0	0
JMT	4	1	3	0	4	1	2	0	3	0	2	1
HK	4	3	3	0	3	0	2	0	2	0	2	0
Totals	.83	24	.87	13	.70	6	.73	2	.58	3	.70	2

Table III. Detection (DET) and false alarm (FA) data for the real-time word spotting system. Under each condition the number of possible detections per talker is 4 in the untrained case and 3 in the trained case. Total elapsed time for all subjects is 1 hour.

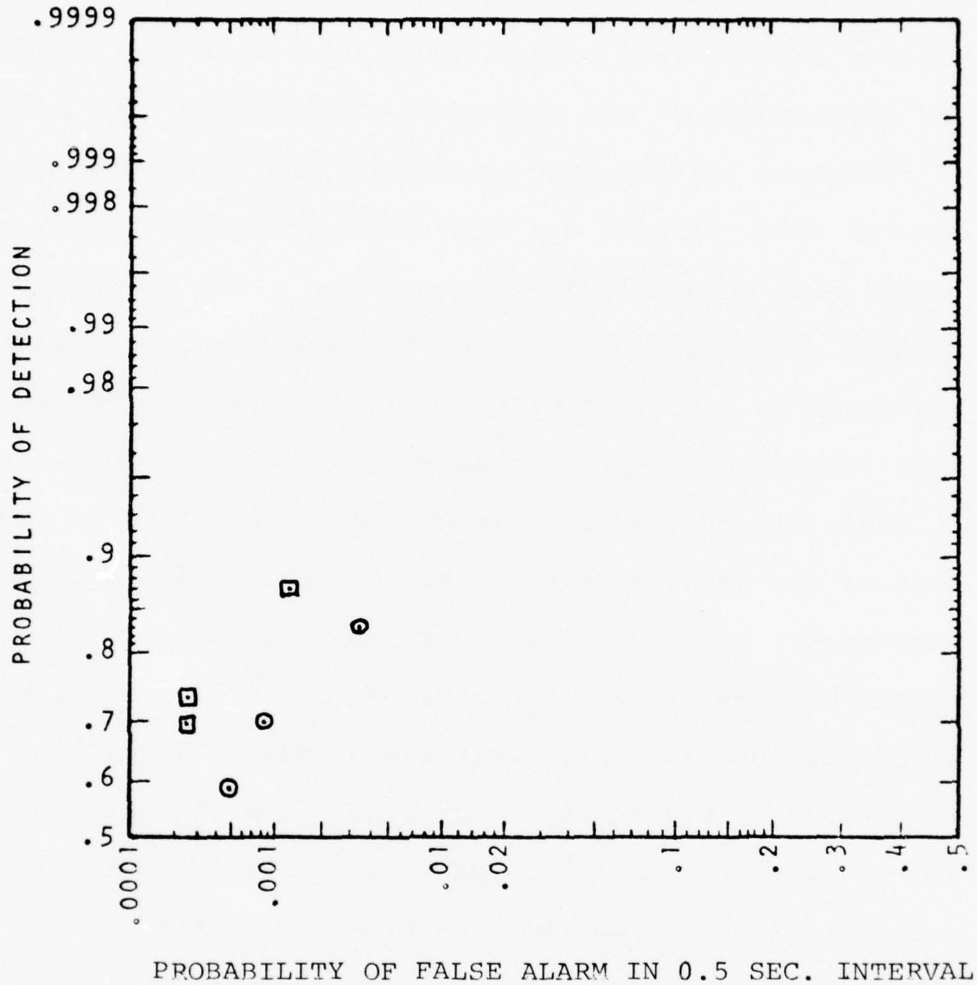


Figure 10. ROC curves for the continuous speech word spotting algorithm derived from the data in Table III.

Earlier tests of a 25-subject telephone speech data base yielded intermediate scores, consistent with the idea that the performance of the system is a function of the number of different talkers used to compile the reference patterns. Several other phrases and data bases of female voices were tested with generally similar results. The numerical results presented here are the only ones generated by the algorithm as described with enough data taken under comparable conditions to produce an ROC curve. It can be seen in Table III that a minority of the talkers tend to produce most of the false alarms. This suggests that improved performance might be attained whenever it is possible to select talkers or, as suggested by the earlier results with small populations, limit the number of talkers to be recognized. Figure 10 was produced from the data in Table III by figuring the false rate per 0.5 second-- the largest duration of the target word observed. At the low end of the curve with 2 or 3 false alarms per hour, the error of measurement is large on a relative scale; nevertheless the fit to an unbiased ROC curve for normally distributed data is good.

Under the same conditions the closest-fitting reference patterns for each half-syllable were further trained to each talker's voice by averaging the mean

value statistics with the observed spectral data for the first instance of the target word in the script; then the detections and false alarms were counted for the talker's reading of the entire script, discounting the single training word. The observed data were given a weight of 0.75 in computing the averaged mean value parameters, while the previous values of the parameters were given a weight of 0.25. The variance parameter was left unchanged, as were the parameters of the temporal window masks. The same set of pattern detection thresholds for the likelihood functions was used; thus a new ROC curve was generated under conditions which corresponded exactly with the conditions of the first curve, except for the training of the mean value parameters (top curve of Figure 10). We expected the detection rate to increase greatly, since the patterns were now tuned to the talker; for the same reason the false alarm rate should have remained the same or increased. Surprisingly, the average detection rate increased only slightly, while the false alarm rate significantly decreased. This effect remains unexplained, though we suspect that an interaction between the pattern likelihood detector settings and the optimum parameters for the subsequently applied concatenation rules is responsible.

Applying the confidence level methods to the experimental results, we had for the small data base one missed detection in a total of four trials for each of four subjects, and 6 false alarms per hour. The experimental detection rate is therefore 0.94 and the sample standard deviation S = 0.108. The student's t distribution method gives a lower 90% confidence level Pc = 0.84. The non-parametric method with the binomial distribution yields Pc = 0.50. The presumably more valid results from Table III at 6 false alarms per hour is 0.70. The confidence levels for this latter probability derived from 10 talkers are 0.57 for the t distribution method and 0.45 for the non-parametric method.

Further development effort, particularly on the concatenation rules and implementation of the remaining pieces of the origanally proposed algorithm, may be expected to improve the performance figures. The sharp difference in performance between small and large populations serves as a warning that large data bases are needed to assess talker-independent performance; the probability models we have presented underscore this requirement. These models also suggest that improvement can be made by setting the pattern detection in a forced-choice decision context and by revising the pattern concatenation rules to incorporate

a priori information on the probabilities of alternate syllable "spellings" of the target word.

II SOFTWARE

The following sections are a description and flowchart of the software pertinent to an understanding of the Key word spotting algorithm. The program of interest is SPIN3M, SPeech INterpreter 3, Multiple word spotting.

Functions Of The Program

The functions of SPIN3M, as controlled by the key board and console are:

1. To accept specification of (up to 2) target words, read in the appropriate reference file of statistical data, and to initialize appropriate tables for the search of words. This function has not been flowcharted, but reference is made to its routine, called "SETPTR".

2. To input continuous speech, and search at a real time rate for occurences of either of the two target words. While doing so, information concerning the status of the algorithm is to be output to the key-board, and special identifying symbols are to be output upon successful key word detection. The function is performed by the subroutine SPIN4M, whose flowchart is on flowchart pages 66 through 78.

3. To stop Key word search with preservation of algorithm status and all data for the last 2.56 seconds, at the discretion of the operator, upon a successful detection, a false alarm, or at any other time. This is done as part of SPIN4M, with conditional exits at appropriate algorithm points.

4. To search for words on a non-real time basis, in the 2.56 seconds of speech data stored at the time of the real time search halt. The subroutine SPN4NR performs this function, and is flowcharted on page 79.

5. To calculate on a non-real time basis, for each 10 ms. interval of the last 2.56 seconds, the likelihood that a given pattern existed. This is done by IPART2, found on flowchart page 80. Flowchart pages 81-91 document all other routines necessary for functions 2-5, and are referenced as subroutines on the first 15 pages of the flowchart.

LANGUAGE AND GROSS STRUCTURE OF THE PROGRAM

SPIN3M is written in 3 languages, consequently it may be said that there are 3 levels of control during its execution.

The top level is under FORTRAN control, which executes all operations for which time is no consideration. This includes I/O (except PDP-11 to Vector Processor I/O), and the keyboard interactive command interpreter. After accepting commands from the keyboard, the FORTRAN code calls the necessary PAL subroutines. The FORTRAN routines are not flowcharted.

The middle level of control is PAL, or PDP-11 assembly language code. The PAL code is organized as subroutines which are called to execute the real or non-real time word spotting functions. PAL is used to generate most of the pattern concatenation (pattern sequencing) control logic, to control vector processor operations, and generally to direct the word spotting algorithm. PAL routines are described on flowchart pages 66-82.

Bottom level of control is within the vector processor, as instructed by Vector Computer Assembly

language, or VCASM code. The PAL subroutines direct the relinquishing of bus mastership from the PDP-11 to a special high-speed array processor. This array or vector processor performs fast calculations, facilitating execution of preprocessing, time registration, and sound unit (pattern) similarity computation. During the execution of a vector processor routine, the vector processor may read or write to the PDP-11 memory, starting at the address contained in the vector computer bus address register. Following the completion of a vector processor routine, the vector processor halts and control returns to the PDP-11, resuming the execution of PAL code. The vector processor routines are flowcharted as subroutines on flowchart pages 83-91.

PROGRAM DATA STRUCTURES

All PAL and VCASM variables are integers, with a maximum of 16 and 32 bit resolution respectively. All PDP-11 arrays are composed of 16 bit integers. The arrays important to the key word spotting algorithm may be categorized into two types: buffers for input data and arrays of reference and status data.

The contents of the input data buffers may change with every new input frame, and those that accumulate data over a number of frames must be circularized to avoid an attempt to store data beyond their upper boundary. By "circularized", the following is meant. Each time new data is added to the present buffer contents, the buffer pointers are advanced to the destination of the next datum, until they point past the end of the buffer. At this time the destination pointer is reset to the start of the buffer and old data is overwritten by the next input frame. This circular data storing technique implies that real time input data is retained for only a brief interval, during which all processing and decision making must be done. Based on considerations of space limitations and algorithm performance, all input data buffers necessary to real time key word spotting have

been circularized to a length corresponding to 2.56 seconds of input data, or 256 input frames.

Every 10 milliseconds a new "frame" is generated by the hardware autocorrelator, and preprocessed by the vector processor. The results of preprocessing are 3 data elements: a spectrum, the frame's subjective time, and the frame's amplitude. The 32 point smoothed, equalized, and log-transformed spectrum is calculated and stored in the frame array JIN, as 32 consecutive 16 bit words. Thus JIN is a circular buffer of spectrum frames, in temporal order, with one frame starting every 64 bytes. The offset to the destination of the next spectrum frame is contained in JINOFS. The circularization of JIN is accomplished by making JINOFS a modulo 16384 offset, that is, modulo 256x64. The frame's subjective time is a 16 bit precision integer, and is stored in 2 bytes of the array JARC. The offset to the destination of the next frame's subjective time is contained in JARCOF, which is made modulo 512 = 256x2 to circularize JARC to a length of 256 two byte subjective times. The amplitude of the frame is initially output by the vector processor as a 16 bit precision integer in the final word of the 32 word spectrum. In this manner it is used by the likelihood routines as a

parameter equivalent in importance to any spectrum point. When real time analysis is halted, all the amplitudes are stored in one buffer, to facilitate non-real time analysis. This buffer is called JAMP, has a length of 512 bytes, and is not circularized since it has no real-time word spotting application.

Every 10 milliseconds, after preprocessing, a new pattern is designated as a combination of three previous input frames. The pattern designated is associated with the frame that was input 31 frames ago. The designation of the pattern associated with a given time involves the specification of pointers to the three frames of the pattern. These pointers are stored in the 3 picked frame buffers FRAM1, FRAM2, and FRAM3. Only the past 256 patterns are valid because data older than 2.56 seconds is lost, thus any pointers to that data which designate a pattern, are meaningless. Since the pointers are 16 bit precision integers, the construction of FRAM1, FRAM2, and FRAM3 is identical to that of JARC, with the offset to the pointer's destination corresponding to a time 31 frames behind the current frame time (see flowchart page 68). In summary, there are five input data buffers used in real time key word spotting, each updated

and circularized to a length of 256 entries every 10 milliseconds. See the table at the end of this section for a summary of input data buffer utilization.

The remaining arrays that are important to real time key word spotting may be categorized as containing either reference or status information. These arrays, once initialized at the beginning of the word spotting run remain either unchanged or are only slightly modified. The arrays in this category are IFILT, IPSTAR, IWDSYM, IWHS, and IWDAN, and the Word Descriptor Arrays, contained in the module WORDS.

IFILT is the cosine transform matrix used to compute the spectrum of the input data. This array remains constant. IPSTAR is a workspace used by the program when calculating the prosodic timing characteristics of a spotted word.

When key word spotting initialization is executed, the target words are set. Associated with each target is a symbol, a Word Descriptor Array, and mean value and standard deviation statistics. For the Kth target word, the Kth element of IWDSYM is the associated symbol and the Kth element of IWDAN is a pointer to the associated Word Descriptor Array. Thus IWDSYM is an array of target word

symbols, and IWDAN an array of target word Word Descriptor Array pointers. The mean value and standard deviation statistics for each pattern of all legitimate target words are read into the array IWHS. This also is done at initialization. IWHS remains constant until the operator chooses to introduce a new set of statistics. The SETPTR subroutine assures that statistics for all specified target words may be found in IWHS. It then sets pointers to the statistics for each pattern of every target word, in the target word's Word Descriptor Array (WDA). Once this is done a complete definition of each target word may be found in its WDA, and all references to the relevant statistics in IWHS are made through the pointers in the WDA.

Basic to an understanding of the program strategy is an understanding of the structure of the Word Descriptor Array. After initialization this array contains a complete description of the target word's patterns and timing, and all the necessary information concerning the status of the search for this word (e.g., how many patterns detected so far, etc.). The use of the WDA allows the searches for multiple target words to be independent and asynchronous. All information about algorithm status exterior to the WDAs is target word independent.

The Word Descriptor Array is organized into three sections: first the header, primarily containing information on the history and status of the search, then a series of pattern parameter lists yielding an exact description of all the patterns, their alternatives, and the interpattern timing, and finally two arrays used for the prosodic timing tests which follow the detection of the whole word pattern sequence.

The WDA header is presently 24 words long, but is structured for easy extensibility. Associated with each target is an "analysis time", which indicates how much of the data presently in the input data buffers has been searched. Analysis time is in the same units as what we refer to as "real time", that is, one unit corresponds to one new frame, or 10 milliseconds by the clock. For every new input frame, the current real time is incremented, and for every frame in the buffers of past data which is processed and searched, the analysis time is incremented. Each target word has its own analysis time in its WDA, thus the search for one target word may be 50 frames behind current real time, while the search for another is 100 frames behind. Analysis time is of course never ahead of current real time, but is also never allowed to fall far behind current real time, because that would imply

analysis of lost data. The target word's header contains the analysis time associated with that target word, called "T", and the corresponding "analysis" subjective time array offset "JARCON". When a pattern is detected, the logic updating the header notes this by incrementing the count of patterns detected, saving the real time of the pattern likelihood peak, saving the likelihood at that peak, and saving offsets to the subjective time and frame of the peak. Various timers are also set to force timing constraints on the detection of the next pattern. The header also is set to point a new pattern parameter list, in section 2 of the WDA, designating it as the current target pattern. See diagram for an exact description of the WDA header.

The WDA pattern parameter lists represent each of the alternative patterns comprising the word pattern sequence. These lists are linked to one another, in that each pattern contains a pointer to its alternative pattern parameter list if one exists, and a pointer to its succeeding pattern parameter list if it is not the final pattern in the word pattern sequence. The pattern parameter list also contains pointers to the statistics for that pattern, and real time timing constraints for the detection of the succeeding pattern.

Following the pattern parameter lists are 2 arrays used for prosodic timing parameter calculation and testing. The first is an array of the maximum and minimum allowable values for each of the 2n-1 prosodic timing parameters in an n pattern word. The second array is an n word buffer meant to contain the pattern likelihood peak time for each pattern in the word pattern sequence. It is filled during the word spotting routine run. These peak times are used as raw data for the prosodic timing parameter calculation and testing routine described on flowchart pages 76 and 77.

A detailed depiction of the Word Descriptor Array contents and their ordering follows:

SPIN4M (MULTIPLE WORD SEEKING ALGORITHM)

WORD DESCRIPTOR ARRAY K (FOR WORD W/ SERIAL #K)

SET WDAPTR = IWDAN(K) "HEADER" INFO:

IWDAN(K)	PTR TO 1ST PATTERN PARAMETER LIST
2(WDAPTR)	/WORD SPELLED OUT
4(WDAPTR)	IN UP TO SIX
6(WDAPTR)	ASCII CHARACTERS/
1Ø(WDAPTR)	= CURPAT (WDAPTR) ADDRESS OF PATTERN PARAMETER LIST FOR CURRENT PATTERN SOUGHT
12(WDAPTR)	= SUMPAT (WDAPTR) CUMULATIVE # OF PATTERNS DETECTED SO FAR (FOR THIS WORD)
14(WDAPTR)	= T1 (WDAPTR) = TP + WINDOW = EXPIRATION TIME OF CURRENT PATTERN SEARCH
16(WDAPTR)	UNUSED
2Ø(WDAPTR)	= WDPCNT (WDAPTR) # OF PATTERNS COMPRISING THE WORD
22(WDAPTR)	= TIMER (WDAPTR) = TP_1 + WDLMIN = EARLIEST ACCEPTABLE TIME FOR TOTAL WORD END
24(WDAPTR)	= WDSTRT (WDAPTR) FLAG SET IF WORD STARTED
26(WDAPTR)	= PASTRTT (WDAPTR) FLAG SET IF PATTERN STARTED
3Ø(WDAPTR)	= TØ (WDAPTR) = (T OF 1ST THRESH CROSSING) + #TRKTIM = EXPIRATION TIME OF PEAK TRACKING FOR THIS PATTERN
32(WDAPTR)	= TP (WDAPTR) TIME OF LAST LIKELIHOOD PEAK FOR CURRENT PATTERN
34(WDAPTR)	= MAXL (WDAPTR) LIKELIHOOD VALUE OF LAST LIKELIHOOD PEAK FOR CURRENT PATTERN
36(WDAPTR)	= UNUSED
4Ø(WDAPTR)	= PTMAR (WDAPTR) POINTER TO PROSODIC TIMING MAXIMUM AND MINIMUM ARRAY
42(WDAPTR)	= IPTRT (WDAPTR) POINTER TO BE STEPPED THROUGH PEAK TIME ARRAY, INITIALLY = IPATIM
44(WDAPTR)	= IPATIM (WDAPTR) POINTER TO PEAK TIME ARRAY FOR THIS WORD (FOLLOWS PTMAR)

46(WDAPTR)	= T (WDAPTR) ANALYSIS TIME FOR THIS WORD
5Ø(WDAPTR)	= JARCON (WDAPTR) POINTER TO CORRESPONDING "ANALYSIS" SUBJECTIVE TIME IN JARC
52(WDAPTR)	= UNUSED
54(WDAPTR)	= JARCON (WDAPTR) POINTER TO SUBJECTIVE TIME OF LAST LIKELIHOOD PEAK
56(WDAPTR)	= WDLIM (WDAPTR) MINIMUM WORD DURATION

WORD DESCRIPTOR ARRAY: PATTERN PARAMETER LISTS
(FOLLOWS "HEADER" INFO)

PAT1:		# OF THIS PATTERN
PAT1+2:	= THRESH (PAT1)	LIKELIHOOD THRESHOLD SETTING FOR THIS PATTERN
" +4:	= WINDOW (")	MAXIMUM DURATION OF SEARCH FOR NEXT PATTERN (= 0 FOR LAST PATTERN)
" +6:		# OF FRAMES AFTER PEAK OF THIS PATTERN DURING WHICH NEXT PATTERN MAY NOT BE FOUND
" +10:	= ALTPAT (")	ADDRESS OF ALTERNATE PATTERN PARAMETER LIST
" +12:	= NXTPAT (")	ADDRESS OF NEXT PATTERN PARAMETER LIST
" +14:	= MEANS (")	POINTER TO MEAN STATISTICS FOR THIS PATTERN
" +16:	= STDS (")	POINTER TO STANDARD DEVIATION STATISTICS FOR THIS PATTERN
PAT2:		
	. (SAME AS ABOVE)	
	.	
	.	

PATN:	N	NUMBER OF THIS PATTERN
	= THRESH (PATN)	
	= WDTIME (")	0 (THIS IS THE LAST PATTERN IN WORD PATTERN SEQUENCE)
		0
	= ALTPAT (")	0
	= NXTPAT (")	0
	= MEANS (")	POINTER TO MEANS FOR THIS PATTERN
	= STDS (")	POINTER TO STANDARD DEVIATIONS STATISTICS FOR THIS PATTERN

WORD DESCRIPTOR ARRAY: PROSODIC TIMING MAX AND MIN ARRAY
(FOLLOWS PATTERN PARAMETER LISTS)

ASSUME THIS IS THE WDA FOR WORDS WITH SERIAL # =N, WDPCNT = K

PTIMN:	MAX FOR SCALED PEAK TIME 1
"+2:	MIN " " " " "
"+4:	MAX " " " " 2
"+6:	MIN " " " " "
.	.
.	.
.	.

PTIMN+4 (K-1):	MAX FOR SCALED PEAK TIME K
	MIN " " " " "
	MAX FOR SCALED DURATION OF PAT. 1
	MIN " " " " " "
	MAX " " " " " 2
	MIN " " " " " "
PTIMN+4 (2K-±0:	MAX FOR SCALED DURATION OF PAT. K-1
	MIN " " " " " ."

WORD DESCRIPTOR ARRAY: PATTERN LIKELIHOOD PEAK TIME ARRAY
(FOLLOWS ABOVE PROSODIC TIMING CONSTRAINTS)

PKARN:	TIME OF PEAK 1
	.
	.
	.
	TIME OF PEAK K

END OF WORD DESCRIPTOR

FLOWCHART VARIABLE NAMING CONVENTIONS

Due in part to the fact that SPIN3M is written in 3 languages, the flowchart conventions for variable naming require clarification. Consider the name "X". If X is referenced alone, the variable named has the value of the word at address X. #X is the variable whose value is the address X. Thus #JARC is the address of JARC. @X refers to the variable whose address is contained at address X. This is "indirect" addressing, and according to PAL conventions X must be a register. X(Rn) is the variable whose address is the address X plus the contents of Rn where Rn is register n. All of the preceeding are essentially PAL conventions. X subscripted by an italic character "*i*" implies that X is the name of an array, and that the variable referenced is the *i*th element of the array. This is a FORTRAN convention. If X is the character "A" or "B", then An or Bn where n is a positive integer (less than 256) refers to the nth word of the vector computer A or B memory, respectively. A Greek subscript to X (usually upper or lower case sigma in the flowcharts) implies that the variable referenced is an element of a Word Descriptor Array or WDA (see dictionary of terms). An upper case sigma (Σ) indicates that the variable referenced is part of the header information of the Σth WDA.

A lower case sigma (σ) indicates that the variable referenced is an element of the σth pattern parameter list of the Σth Word Descriptor Array. For clarification of WDA structure see data structures sections on page 58. When it is used in this context X names the element of the array referenced and Σ or σ denotes the array. Thus the address of the variable $CURPAT_{\Sigma}$ would be the starting address of the Σth WDA plus an offset equal to the predefined value of CURPAT. The address of $NXTPAT_{\sigma}$ would be the starting address of the σth pattern parameter list in the Σth WDA plus an offset equal to the value of NXTPAT. In short, all Greek subscripted names are indices to some part of a Word Descriptor Array, determining which element is the variable referenced. For a summary of variable naming conventions see chart below.

Summary of Flowchart Variable Naming Conventions

Name	Variable Named
X	Variable whose address is X
#X	Variable whose value is address X
@X	Variable whose address is at address X
X(Rn)	Variable whose address is X plus contents of register n
X_i	ith element of array X
An or Bn	nth word of V.C. A or B memory
X_{Σ}	Variable found X bytes after start of Σth Word Descriptor Array
X_{σ}	Variable found X bytes after start of σth pattern parameter list in Σth WDA (X has been assigned a numerical value in above 2 cases)

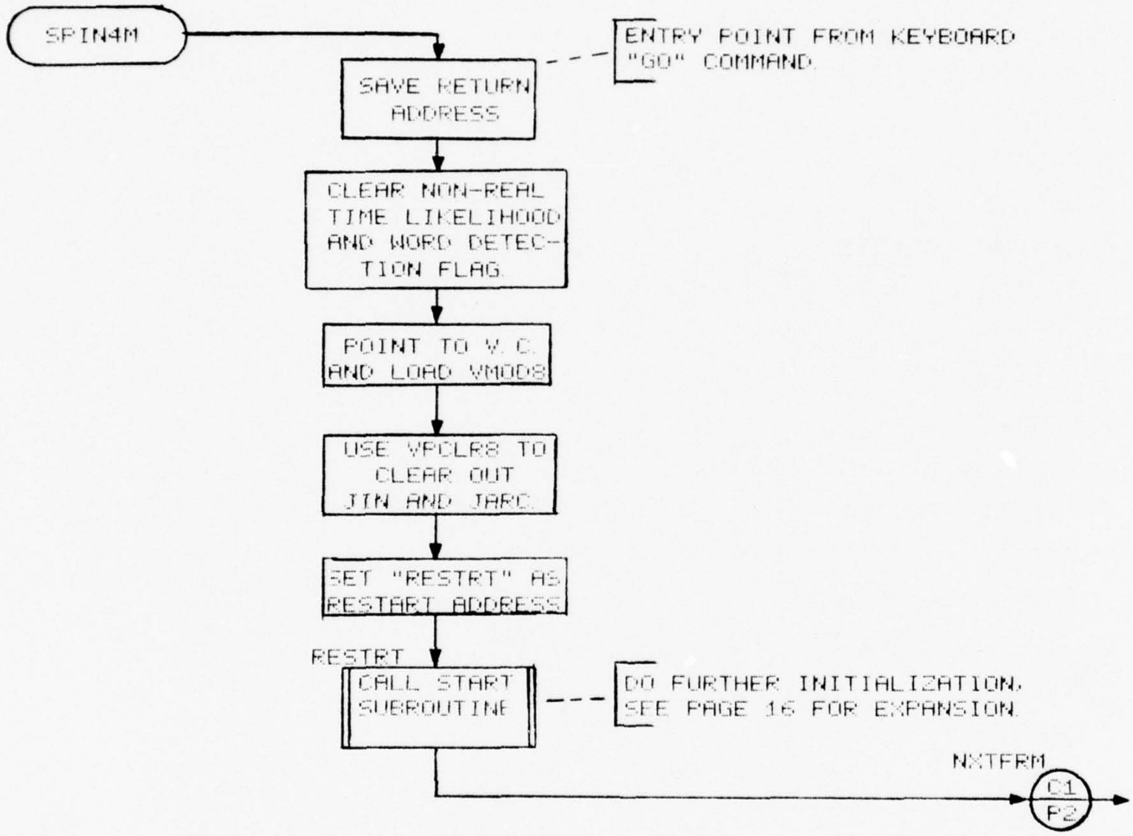
SUMMARY OF BUFFER UTILIZATION

Buffer	Use
JIN	Stores 32 word spectrum frames (circular)
JARC	Stores 1 word subjective times for each JIN frame (circular)
Jamp	Filled with amplitude of each frame in JIN
FRAM1	Pointers to first picked frame of each pattern
FRAM2	Pointers to 2nd picked frame of each pattern
FRAM3	Pointers to 3rd picked frame of each pattern
IFILT	Cosine transform matrix
IPSTAR	Prosodic timing test workspace
IWDSYM	Array of symbols associated with each target word
IWHS	Statistics for all legitimate target words
IWDAN	Array of pointers to target word WDAs
WDA	A Word Descriptor Array, reference and status information unique to one target word

FLOWCHART TABLE OF CONTENTS

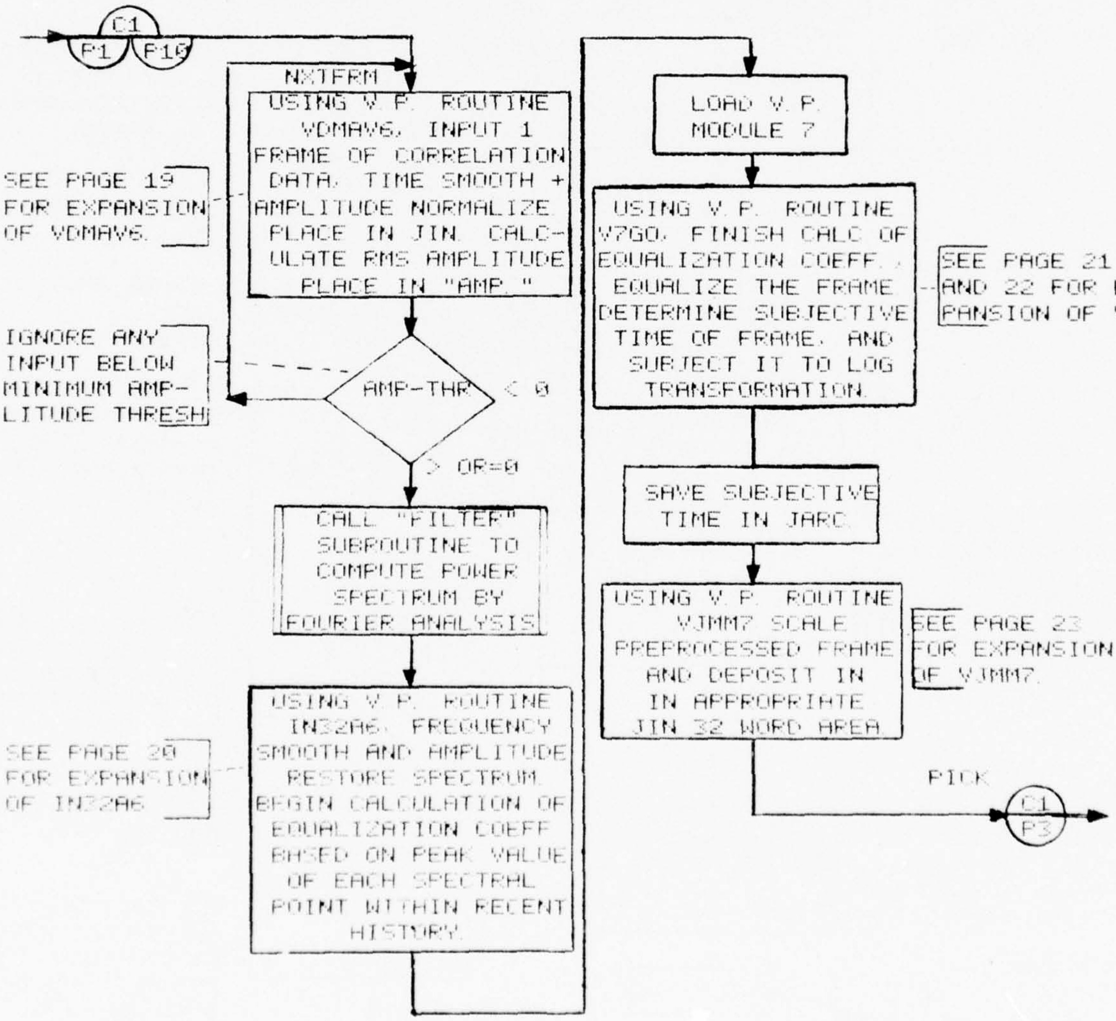
SPIN3M

THIS ROUTINE IS THE REAL TIME KEY-WORD SPOTTER. WRITTEN IN ASSEMBLY LANGUAGE, IT IS CALLED FROM A FORTRAN KEYBOARD INTERACTIVE INTERPRETER. AT TIME OF CALL TO SPIN4M, A TABLE OF POINTERS TO TARGET WORD DESCRIPTOR ARRAYS (IWDAN) MUST BE SET UP, POINTERS TO STATISTICS FOR EACH PATTERN MUST BE SET IN ACTIVE WORD DESCRIPTOR ARRAYS, AND NWORDS MUST BE SET TO # OF WORDS TO BE SOUGHT (MAX OF 2 PRESENTLY).

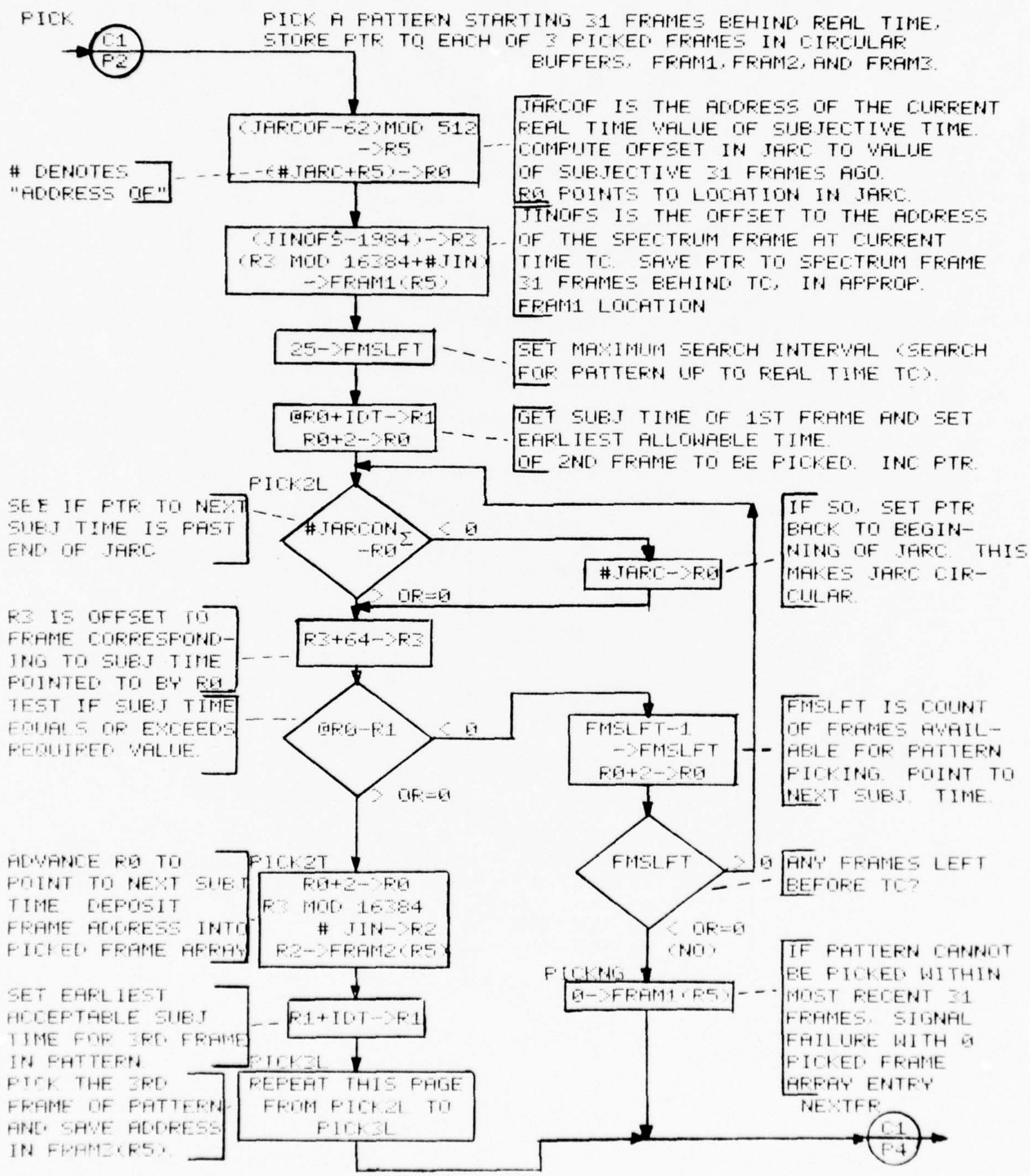


• SPIN3M

CONTROL MUST RETURN HERE FOR EVERY REAL TIME INPUT FRAME, I.E., EVERY 10 MS TO GET NEW FRAME FROM HARDWARE AUTOCORRELATOR. NXTFRM PERFORMS PREPROCESSING TO YIELD TIME SMOOTHED, EQUALIZED, AND LOG-TRANSFORMED 32 WORD SPECTRA IN THE "JIN" BUFFER, WITH THE 32ND WORD CONTAINING THE AMPLITUDE. ALSO PRODUCED IS A BUFFER OF SUBJECTIVE TIME FOR EACH FRAME, "JARC." AFTER NXTFRM PREPROCESSING, "PICK" ROUTINE CHOOSES 3 FRAMES TO FORM A PATTERN, AND 3 LOOPS ARE MADE THROUGH LIKELIHOOD CALCULATING AND PATTERN SEQUENCE TRACKING LOGIC FROM "NEXTFR" TO JUST BEFORE "RETURN." "RETURN" SETS UP TO RETURN CONTROL BACK HERE TO NXTFRM.

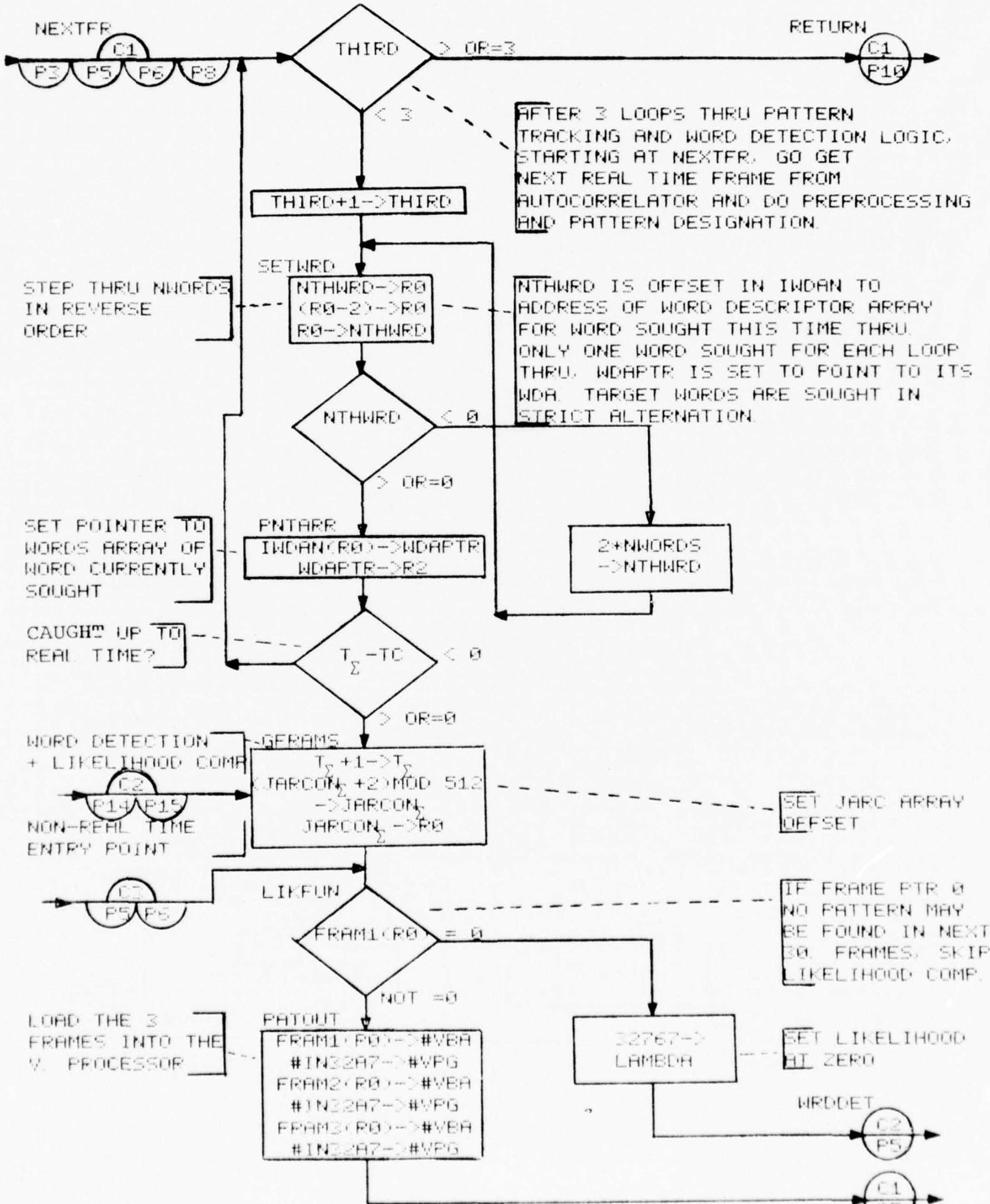


SPIN3M

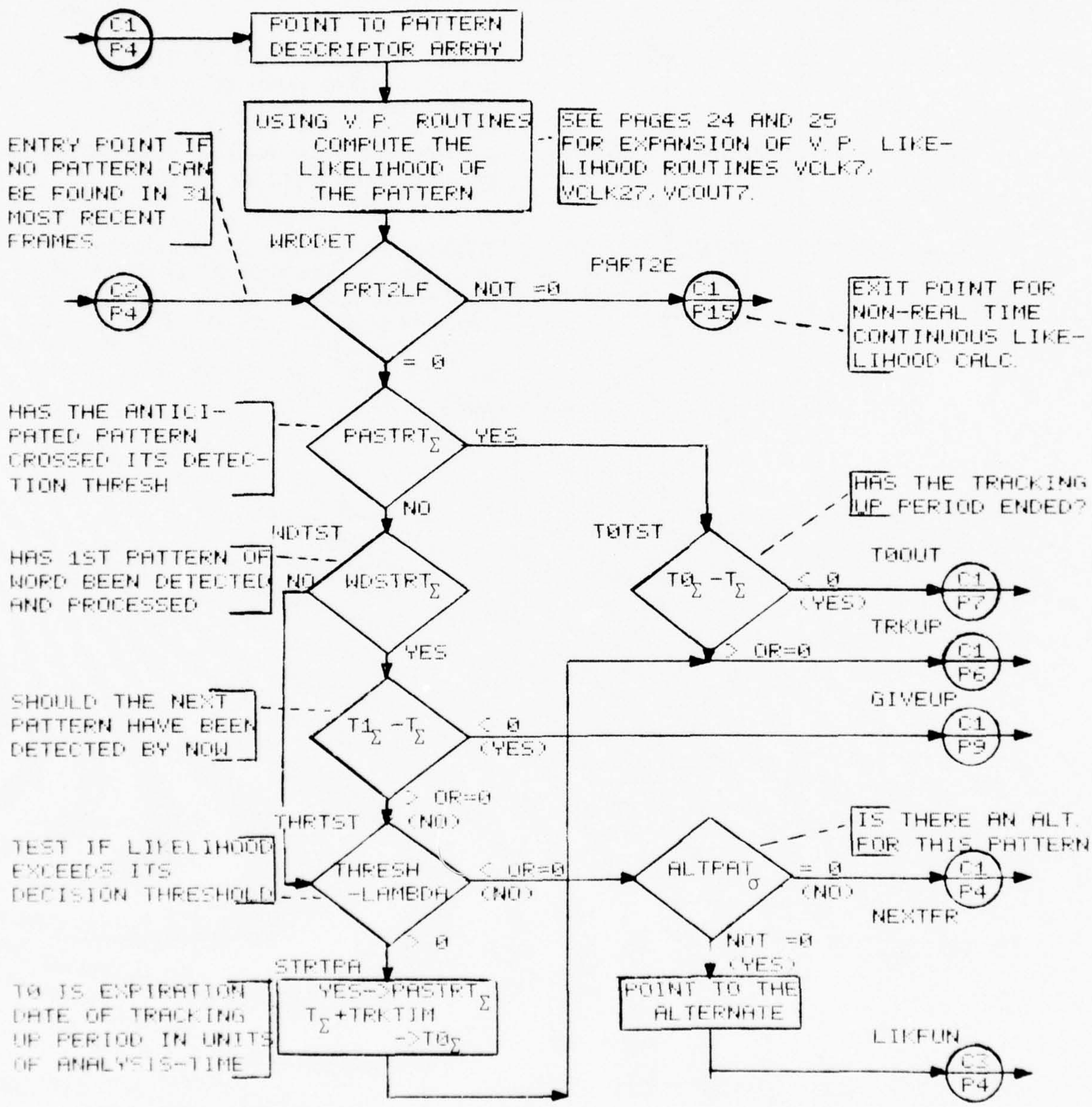


PICK
C1
P2
PICK A PATTERN STARTING 31 FRAMES BEHIND REAL TIME, STORE PTR TO EACH OF 3 PICKED FRAMES IN CIRCULAR BUFFERS, FRAM1, FRAM2, AND FRAM3.
(JARCOF-62)MOD 512 ->R5
-(#JARC+R5)->R0
# DENOTES "ADDRESS OF"
JARCOF IS THE ADDRESS OF THE CURRENT REAL TIME VALUE OF SUBJECTIVE TIME. COMPUTE OFFSET IN JARC TO VALUE OF SUBJECTIVE 31 FRAMES AGO. R0 POINTS TO LOCATION IN JARC.
(JINOFS-1984)->R3
(R3 MOD 16384+#JIN) ->FRAM1(R5)
JINOFS IS THE OFFSET TO THE ADDRESS OF THE SPECTRUM FRAME AT CURRENT TIME TC. SAVE PTR TO SPECTRUM FRAME 31 FRAMES BEHIND TC, IN APPROP. FRAM1 LOCATION
25->FMSLFT
SET MAXIMUM SEARCH INTERVAL (SEARCH FOR PATTERN UP TO REAL TIME TC).
@R0+IDT->R1
R0+2->R0
GET SUBJ TIME OF 1ST FRAME AND SET EARLIEST ALLOWABLE TIME. OF 2ND FRAME TO BE PICKED. INC PTR.
PICK2L
SEE IF PTR TO NEXT SUBJ TIME IS PAST END OF JARC
#JARCON -R0
< 0
#JARC->R0
IF SO, SET PTR BACK TO BEGINNING OF JARC. THIS MAKES JARC CIRCULAR.
> OR=0
R3 IS OFFSET TO FRAME CORRESPONDING TO SUBJ TIME POINTED TO BY R0
R3+64->R3
TEST IF SUBJ TIME EQUALS OR EXCEEDS REQUIRED VALUE.
@R0-R1
< 0
FMSLFT-1 ->FMSLFT
R0+2->R0
FMSLFT IS COUNT OF FRAMES AVAILABLE FOR PATTERN PICKING. POINT TO NEXT SUBJ. TIME.
> OR=0
ADVANCE R0 TO POINT TO NEXT SUBJ TIME. DEPOSIT FRAME ADDRESS INTO PICKED FRAME ARRAY
PICK2T
R0+2->R0
R3 MOD 16384
JIN->R2
R2->FRAM2(R5)
FMSLFT
> 0
ANY FRAMES LEFT BEFORE TC?
< OR=0
(NO)
PICKNG
0->FRAM1(R5)
IF PATTERN CANNOT BE PICKED WITHIN MOST RECENT 31 FRAMES, SIGNAL FAILURE WITH 0 PICKED FRAME ARRAY ENTRY
NEXTFR
SET EARLIEST ACCEPTABLE SUBJ TIME FOR 3RD FRAME IN PATTERN
R1+IDT->R1
PICK3L
PICK THE 3RD FRAME OF PATTERN AND SAVE ADDRESS IN FRAM3(R5).
REPEAT THIS PAGE FROM PICK2L TO PICK3L
C1
P4

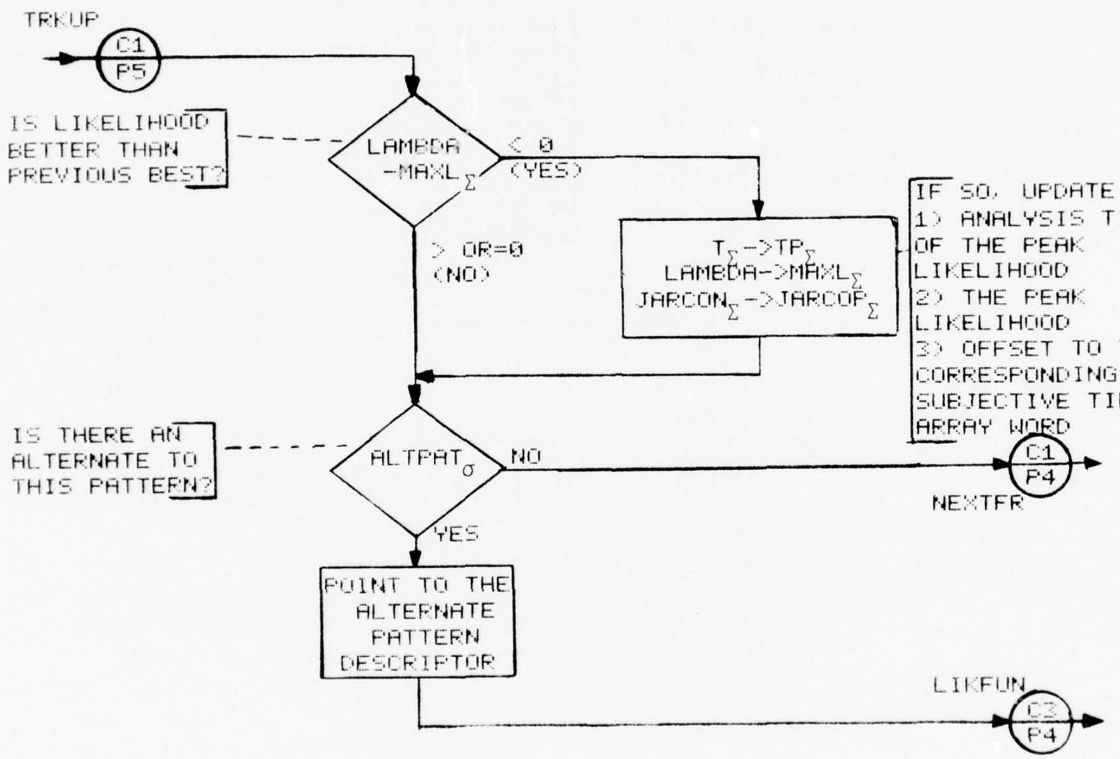
SPIN3M


NEXTFR
C1
P3 P5 P6 P8
THIRD
> OR=3
RETURN
C1
P10
< 3
AFTER 3 LOOPS THRU PATTERN TRACKING AND WORD DETECTION LOGIC, STARTING AT NEXTFR, GO GET NEXT REAL TIME FRAME FROM AUTOCORRELATOR AND DO PREPROCESSING AND PATTERN DESIGNATION.
THIRD+1->THIRD
SETWRD
STEP THRU NWORDS IN REVERSE ORDER
NTHWRD->R0
(R0-2)->R0
R0->NTHWRD
NTHWRD IS OFFSET IN IWDAN TO ADDRESS OF WORD DESCRIPTOR ARRAY FOR WORD SOUGHT THIS TIME THRU. ONLY ONE WORD SOUGHT FOR EACH LOOP THRU. WDAPTR IS SET TO POINT TO ITS WDA. TARGET WORDS ARE SOUGHT IN STRICT ALTERNATION.
NTHWRD
< 0
> OR=0
PNTARR
SET POINTER TO WORDS ARRAY OF WORD CURRENTLY SOUGHT
IWDAN(R0)->WDAPTR
WDAPTR->R2
2+NWORDS
->NTHWRD
CAUGHT UP TO REAL TIME?
T_Σ -TC
< 0
> OR=0
WORD DETECTION + LIKELIHOOD COMP
GFRAMS
T_Σ +1->T_Σ
(JARCON_Σ +2)MOD 512
->JARCON_Σ
JARCON_Σ ->R0
C2
P14 P15
NON-REAL TIME ENTRY POINT
SET JARC ARRAY OFFSET
C1
P5 P5
LIKFUN
FRAM1(R0) = 0
IF FRAME PTR 0 NO PATTERN MAY BE FOUND IN NEXT 30. FRAMES. SKIP LIKELIHOOD COMP.
NOT =0
PATOUT
LOAD THE 3 FRAMES INTO THE V. PROCESSOR
FRAM1(R0)->#VBA
#IN32A7->#VPG
FRAM2(R0)->#VBA
#IN32A7->#VPG
FRAM3(R0)->#VBA
#IN32A7->#VPG
32767->
LAMBDA
SET LIKELIHOOD AT ZERO
WRDDET
C2
P5
C1
P5

SPIN3M

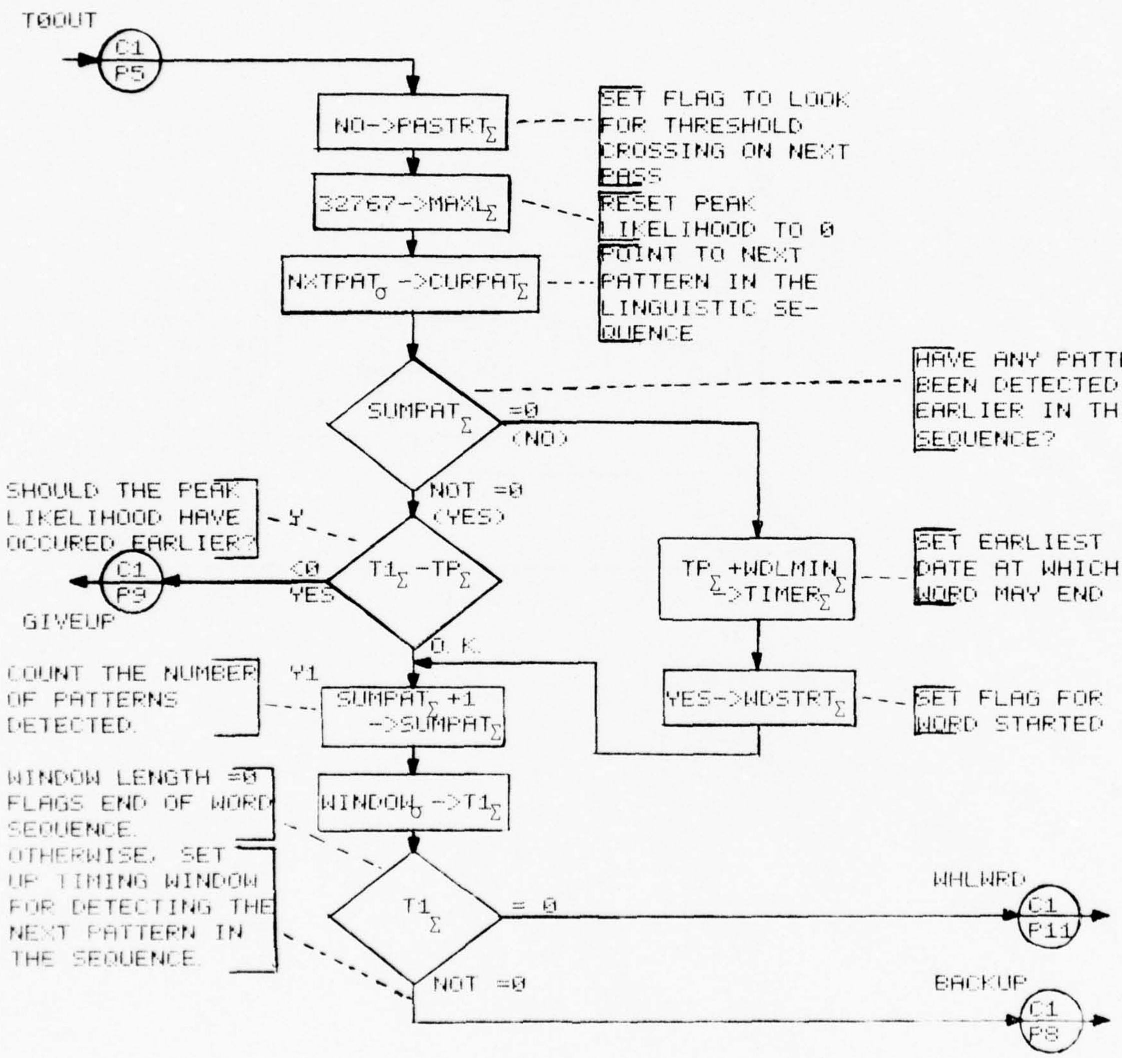

C1
P4
POINT TO PATTERN
DESCRIPTOR ARRAY
USING V.P. ROUTINES
COMPUTE THE
LIKELIHOOD OF
THE PATTERN
SEE PAGES 24 AND 25
FOR EXPANSION OF V.P. LIKE-
LIHOOD ROUTINES VCLK7,
VCLK27, VCOUT7.
ENTRY POINT IF
NO PATTERN CAN
BE FOUND IN 31
MOST RECENT
FRAMES
WRDDET
C2
P4
PRT2LF
NOT =0
PART2E
C1
P15
EXIT POINT FOR
NON-REAL TIME
CONTINUOUS LIKE-
LIHOOD CALC.
= 0
HAS THE ANTICI-
PATED PATTERN
CROSSED ITS DETEC-
TION THRESH
PASTRT
YES
NO
WDTST
HAS 1ST PATTERN OF
WORD BEEN DETECTED
AND PROCESSED
NO
WDSTRT
YES
T0TST
HAS THE TRACKING
UP PERIOD ENDED?
T0 - T
< 0
(YES)
T0OUT
C1
P7
TRKUP
> OR=0
C1
P6
SHOULD THE NEXT
PATTERN HAVE BEEN
DETECTED BY NOW
T1 - T
< 0
(YES)
GIVEUP
C1
P9
> OR=0
(NO)
THRTST
TEST IF LIKELIHOOD
EXCEEDS ITS
DECISION THRESHOLD
THRESH
-LAMBDA
< OR=0
(NO)
ALTPAT
IS THERE AN ALT.
FOR THIS PATTERN
= 0
(NO)
C1
P4
NEXTFR
> 0
NOT =0
(YES)
STRTPA
T0 IS EXPIRATION
DATE OF TRACKING
UP PERIOD IN UNITS
OF ANALYSIS-TIME
YES->PASTRT
T+TRKTIM
->T0
POINT TO THE
ALTERNATE
LIKFUN
C3
P4

SPIN3M



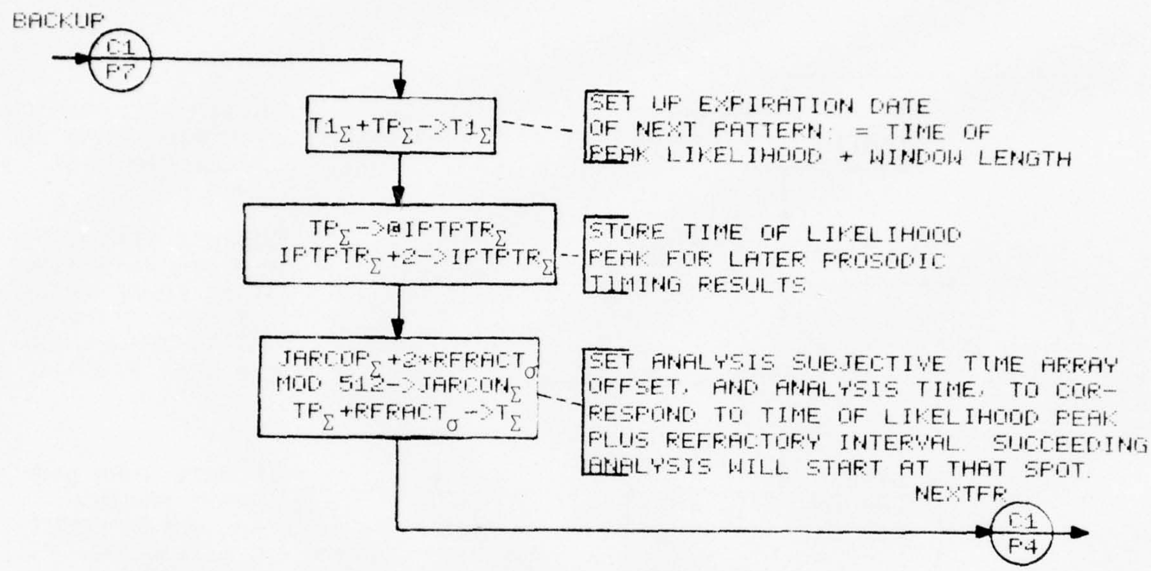
TRKUP
C1
P5
IS LIKELIHOOD BETTER THAN PREVIOUS BEST?
LAMBDA -MAXL$_\Sigma$
< 0 (YES)
> OR=0 (NO)
T$_\Sigma$->TP$_\Sigma$
LAMBDA->MAXL$_\Sigma$
JARCON$_\Sigma$->JARCOP$_\Sigma$
IF SO, UPDATE:
1) ANALYSIS TIME OF THE PEAK LIKELIHOOD
2) THE PEAK LIKELIHOOD
3) OFFSET TO THE CORRESPONDING SUBJECTIVE TIME ARRAY WORD
IS THERE AN ALTERNATE TO THIS PATTERN?
ALTPAT$_\sigma$
NO
C1
P4
NEXTFR
YES
POINT TO THE ALTERNATE PATTERN DESCRIPTOR
LIKFUN
C3
P4

SPIN3M

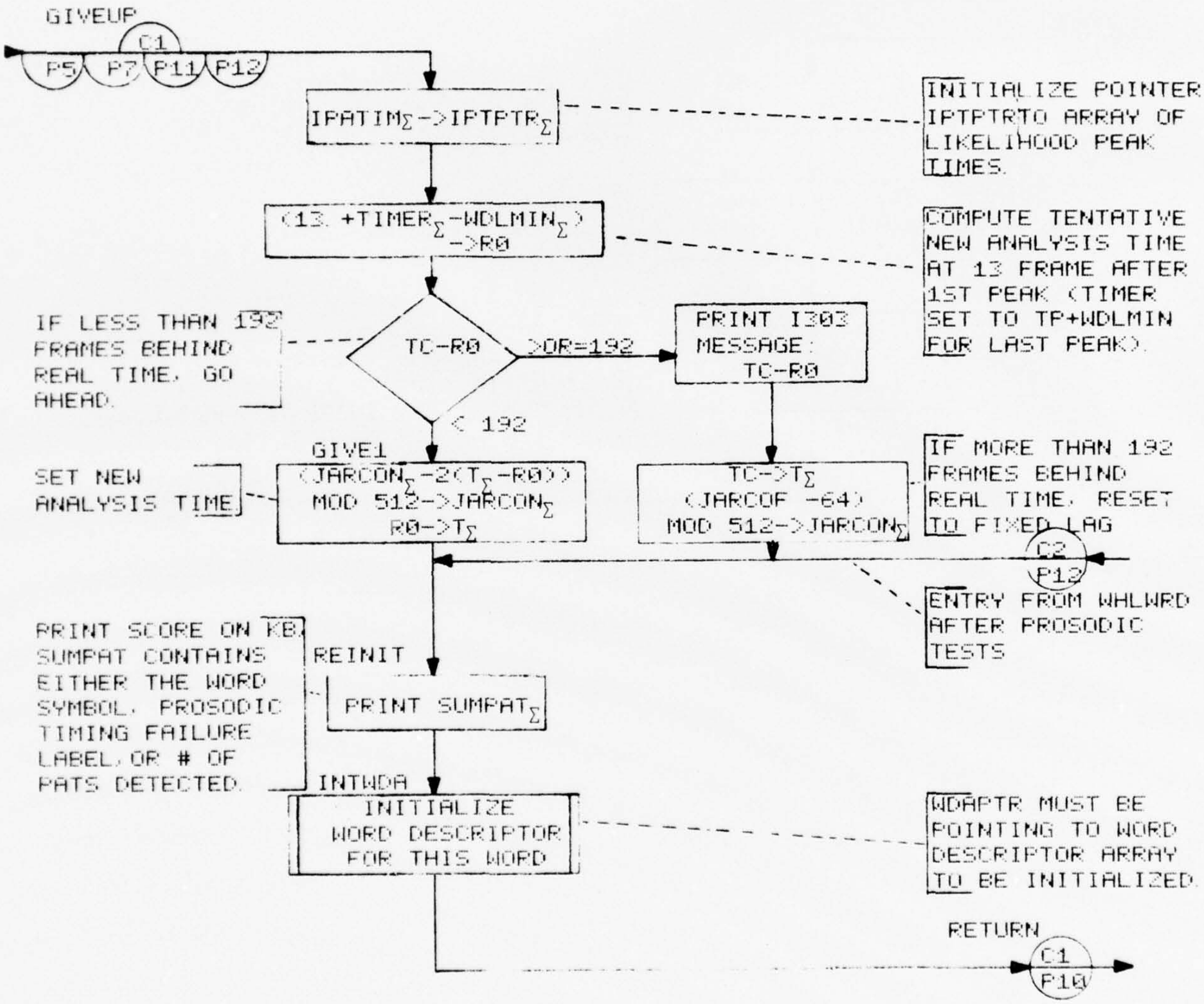


TØOUT
C1 P5
NO->PASTRT
SET FLAG TO LOOK FOR THRESHOLD CROSSING ON NEXT PASS
32767->MAXL
RESET PEAK LIKELIHOOD TO 0
NXTPAT -> CURPAT
POINT TO NEXT PATTERN IN THE LINGUISTIC SE-QUENCE
SUMPAT
=0 (NO)
HAVE ANY PATTERNS BEEN DETECTED EARLIER IN THIS SEQUENCE?
NOT =0 (YES)
SHOULD THE PEAK LIKELIHOOD HAVE OCCURED EARLIER?
Y
T1 -TP
<0 YES
C1 P9
GIVEUP
O. K.
TP +WDLMIN ->TIMER
SET EARLIEST DATE AT WHICH WORD MAY END
YES->WDSTRT
SET FLAG FOR WORD STARTED
COUNT THE NUMBER OF PATTERNS DETECTED.
Y1
SUMPAT +1 ->SUMPAT
WINDOW LENGTH =0 FLAGS END OF WORD SEQUENCE.
WINDOW ->T1
OTHERWISE, SET UP TIMING WINDOW FOR DETECTING THE NEXT PATTERN IN THE SEQUENCE.
T1
= 0
WHLWRD
C1 P11
NOT =0
BACKUP
C1 P8

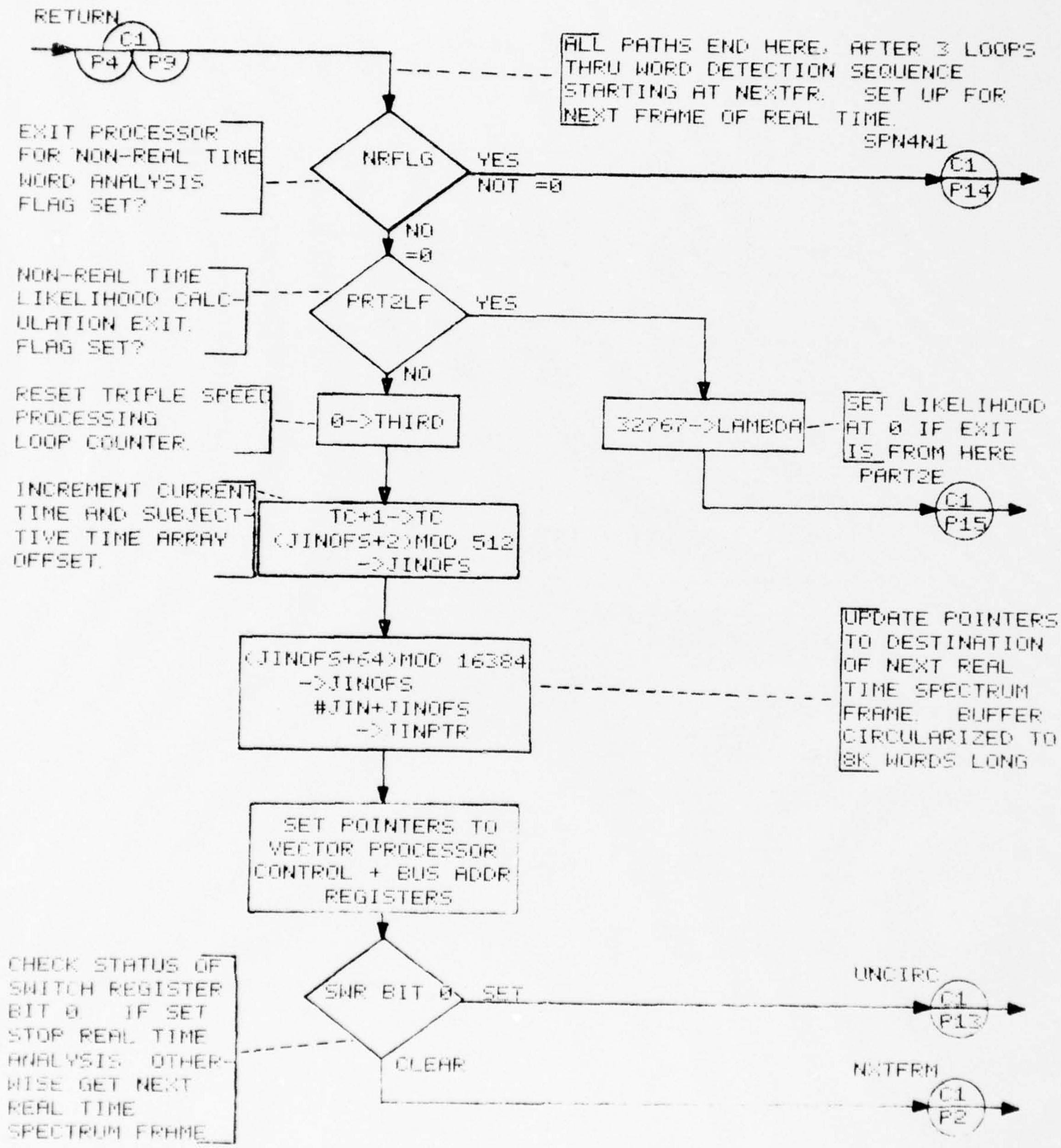
SPIN3M


BACKUP
C1
P7
T1Σ+TPΣ->T1Σ
SET UP EXPIRATION DATE
OF NEXT PATTERN: = TIME OF
PEAK LIKELIHOOD + WINDOW LENGTH
TPΣ->@IPTPTRΣ
IPTPTRΣ+2->IPTPTRΣ
STORE TIME OF LIKELIHOOD
PEAK FOR LATER PROSODIC
TIMING RESULTS
JARCOPΣ+2*RFRACTσ
MOD 512->JARCONΣ
TPΣ+RFRACTσ->TΣ
SET ANALYSIS SUBJECTIVE TIME ARRAY
OFFSET, AND ANALYSIS TIME, TO COR-
RESPOND TO TIME OF LIKELIHOOD PEAK
PLUS REFRACTORY INTERVAL. SUCCEEDING
ANALYSIS WILL START AT THAT SPOT.
NEXTPR
C1
P4

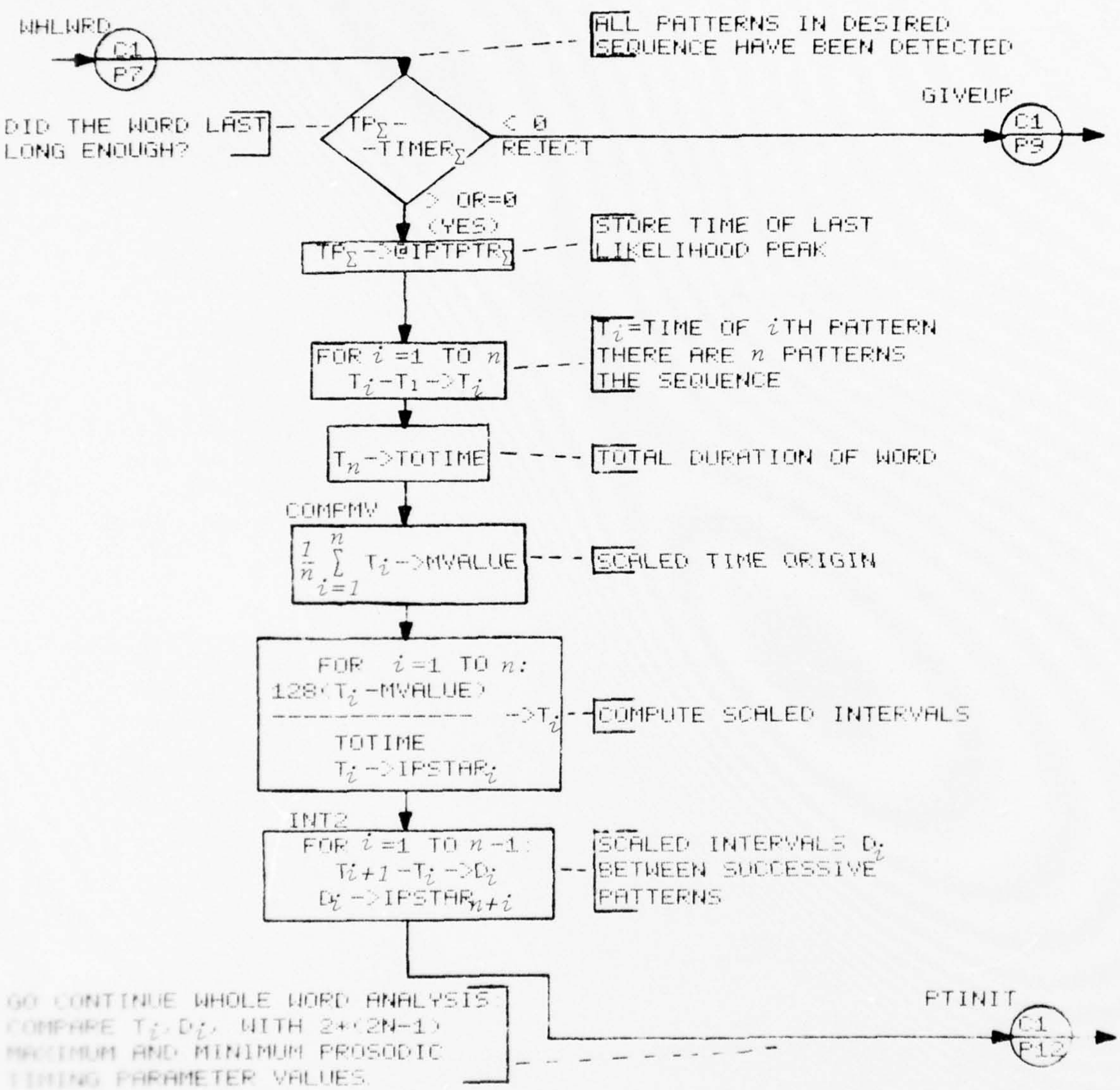
SPIN3M


GIVEUP
C1
P5
P7
P11
P12
IPATIMΣ->IPTPTRΣ
INITIALIZE POINTER IPTPTR TO ARRAY OF LIKELIHOOD PEAK TIMES.
(13. +TIMERΣ-WDLMINΣ) ->R0
COMPUTE TENTATIVE NEW ANALYSIS TIME AT 13 FRAME AFTER 1ST PEAK (TIMER SET TO TP+WDLMIN FOR LAST PEAK).
IF LESS THAN 192 FRAMES BEHIND REAL TIME, GO AHEAD.
TC-R0
>OR=192
PRINT I303 MESSAGE: TC-R0
< 192
GIVE1
SET NEW ANALYSIS TIME
(JARCONΣ-2(TΣ-R0)) MOD 512->JARCONΣ R0->TΣ
TC->TΣ (JARCOF -64) MOD 512->JARCONΣ
IF MORE THAN 192 FRAMES BEHIND REAL TIME, RESET TO FIXED LAG
C2
P12
ENTRY FROM WHLWRD AFTER PROSODIC TESTS
PRINT SCORE ON KB. SUMPAT CONTAINS EITHER THE WORD SYMBOL, PROSODIC TIMING FAILURE LABEL, OR # OF PATS DETECTED.
REINIT
PRINT SUMPATΣ
INTWDA
INITIALIZE WORD DESCRIPTOR FOR THIS WORD
WDAPTR MUST BE POINTING TO WORD DESCRIPTOR ARRAY TO BE INITIALIZED.
RETURN
C1
P10

SPIN3M

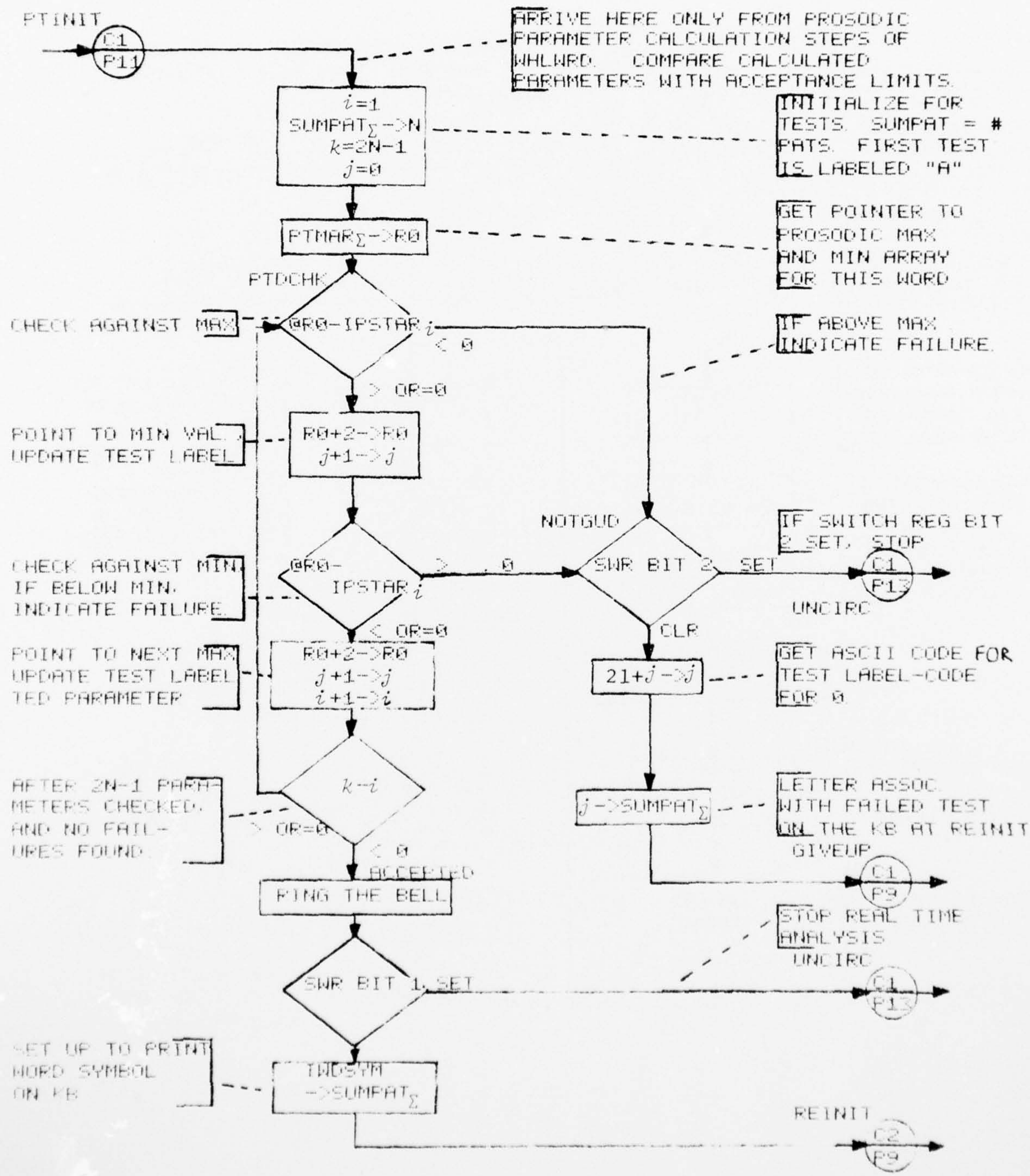

RETURN
C1
P4
P9
ALL PATHS END HERE, AFTER 3 LOOPS THRU WORD DETECTION SEQUENCE STARTING AT NEXTFR. SET UP FOR NEXT FRAME OF REAL TIME.
EXIT PROCESSOR FOR NON-REAL TIME WORD ANALYSIS FLAG SET?
NRFLG
YES NOT =0
SPN4N1
C1
P14
NO =0
NON-REAL TIME LIKELIHOOD CALCULATION EXIT. FLAG SET?
PRT2LF
YES
NO
RESET TRIPLE SPEED PROCESSING LOOP COUNTER.
0->THIRD
32767->LAMBDA
SET LIKELIHOOD AT 0 IF EXIT IS FROM HERE
PART2E
C1
P15
INCREMENT CURRENT TIME AND SUBJECTIVE TIME ARRAY OFFSET.
TC+1->TC
(JINOFS+2)MOD 512
->JINOFS
(JINOFS+64)MOD 16384
->JINOFS
#JIN+JINOFS
->TINPTR
UPDATE POINTERS TO DESTINATION OF NEXT REAL TIME SPECTRUM FRAME. BUFFER CIRCULARIZED TO 8K WORDS LONG
SET POINTERS TO VECTOR PROCESSOR CONTROL + BUS ADDR REGISTERS
CHECK STATUS OF SWITCH REGISTER BIT 0. IF SET STOP REAL TIME ANALYSIS. OTHERWISE GET NEXT REAL TIME SPECTRUM FRAME
SWR BIT 0
SET
UNCIRC
C1
P13
CLEAR
NXTFRM
C1
P2

SPIN3M

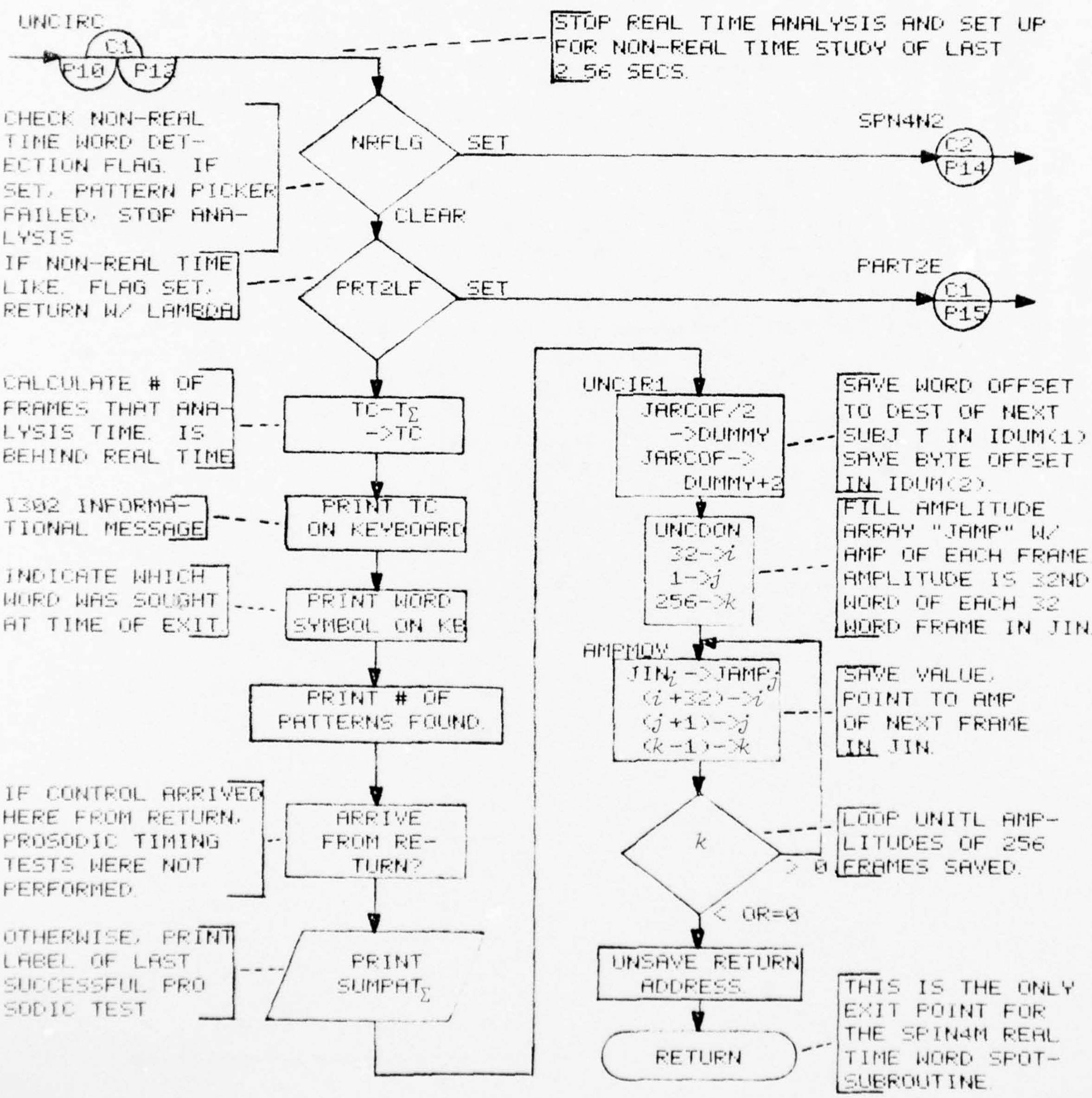


WHLWRD
C1
P7
ALL PATTERNS IN DESIRED SEQUENCE HAVE BEEN DETECTED
DID THE WORD LAST LONG ENOUGH?
TP_Σ − $-TIMER_\Sigma$
< 0 REJECT
GIVEUP
C1
P9
> OR=0 (YES)
TP_Σ ->@IPTPTR$_\Sigma$
STORE TIME OF LAST LIKELIHOOD PEAK
FOR i =1 TO n
$T_i - T_1$ -> T_i
T_i=TIME OF iTH PATTERN
THERE ARE n PATTERNS
THE SEQUENCE
T_n ->TOTIME
TOTAL DURATION OF WORD
COMPMV
$\frac{1}{n}\sum_{i=1}^{n} T_i$ ->MVALUE
SCALED TIME ORIGIN
FOR i=1 TO n:
$\frac{128(T_i - \text{MVALUE})}{\text{TOTIME}}$ -> T_i
T_i ->IPSTAR$_i$
COMPUTE SCALED INTERVALS
INT2
FOR i =1 TO n-1:
$T_{i+1} - T_i$ -> D_i
D_i ->IPSTAR$_{n+i}$
SCALED INTERVALS D_i BETWEEN SUCCESSIVE PATTERNS
GO CONTINUE WHOLE WORD ANALYSIS
COMPARE T_i, D_i, WITH 2*(2N-1)
MAXIMUM AND MINIMUM PROSODIC
TIMING PARAMETER VALUES.
PTINIT
C1
P12

SPIN3M


PTINIT
C1 P11
ARRIVE HERE ONLY FROM PROSODIC PARAMETER CALCULATION STEPS OF WHLWRD. COMPARE CALCULATED PARAMETERS WITH ACCEPTANCE LIMITS.
i=1
SUMPAT∑->N
k=2N-1
j=0
INITIALIZE FOR TESTS. SUMPAT = # PATS. FIRST TEST IS LABELED "A"
PTMAR∑->R0
GET POINTER TO PROSODIC MAX AND MIN ARRAY FOR THIS WORD
PTDCHK
CHECK AGAINST MAX
@R0-IPSTARi
< 0
IF ABOVE MAX INDICATE FAILURE.
> OR=0
POINT TO MIN VAL. UPDATE TEST LABEL
R0+2->R0
j+1->j
NOTGUD
IF SWITCH REG BIT 2 SET, STOP
CHECK AGAINST MIN. IF BELOW MIN. INDICATE FAILURE.
@R0- IPSTARi
> 0
SWR BIT 2
SET
C1 P13
UNCIRC
CLR
< OR=0
POINT TO NEXT MAX UPDATE TEST LABEL TED PARAMETER
R0+2->R0
j+1->j
i+1->i
21+j->j
GET ASCII CODE FOR TEST LABEL-CODE FOR 0
AFTER 2N-1 PARAMETERS CHECKED. AND NO FAILURES FOUND.
k-i
> OR=0
< 0
j->SUMPAT∑
LETTER ASSOC. WITH FAILED TEST ON THE KB AT REINIT
GIVEUP
C1 P9
ACCEPTED
RING THE BELL
STOP REAL TIME ANALYSIS
UNCIRC
SWR BIT 1 SET
C1 P13
SET UP TO PRINT WORD SYMBOL ON KB
IWDSYM ->SUMPAT∑
REINIT
C2 P9

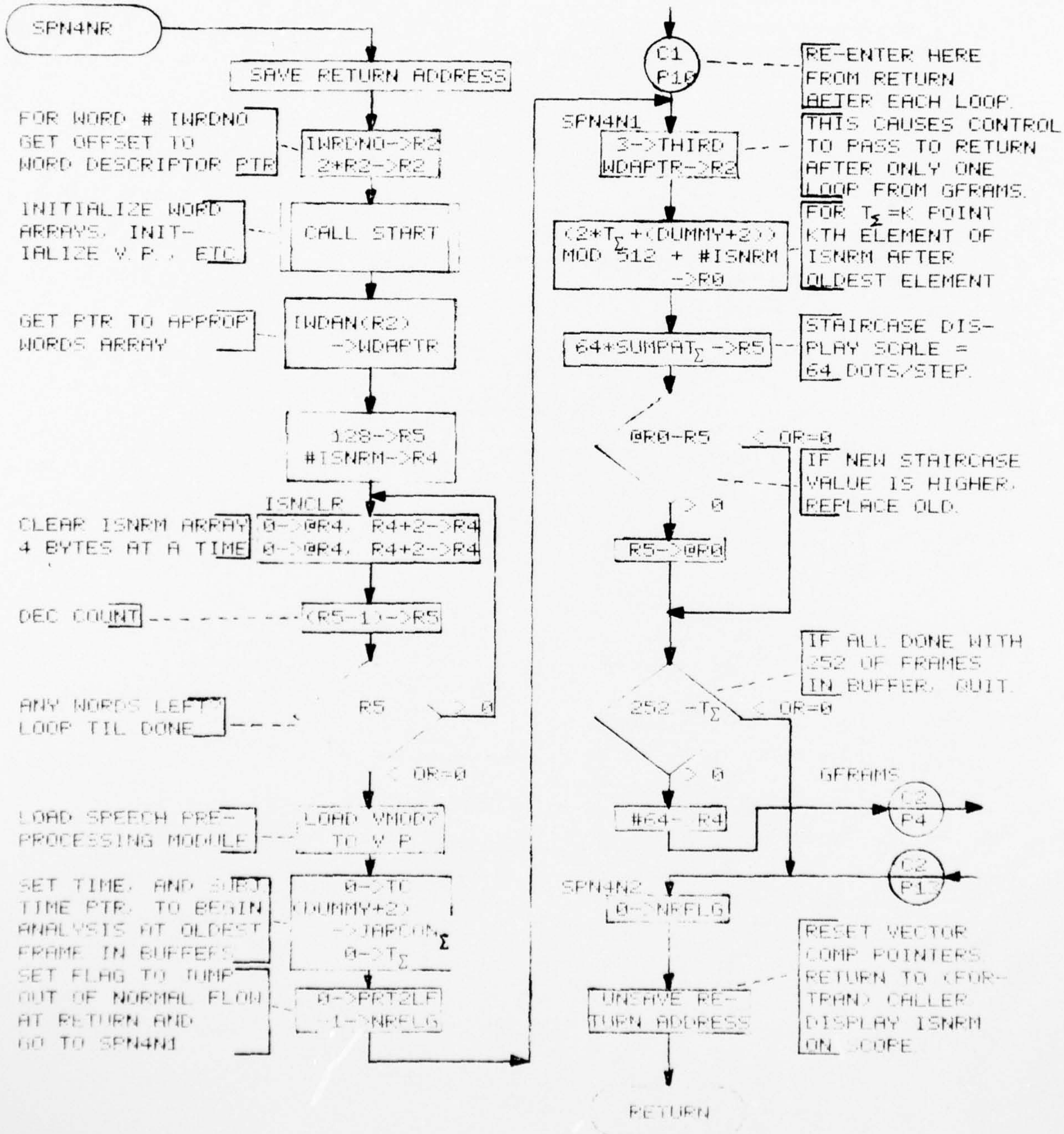
SPIN3M


UNCIRC
C1
P10
P13
STOP REAL TIME ANALYSIS AND SET UP FOR NON-REAL TIME STUDY OF LAST 2.56 SECS.
CHECK NON-REAL TIME WORD DETECTION FLAG. IF SET, PATTERN PICKER FAILED, STOP ANALYSIS
NRFLG
SET
CLEAR
SPN4N2
C2
P14
IF NON-REAL TIME LIKE. FLAG SET, RETURN W/ LAMBDA
PRT2LF
SET
PART2E
C1
P15
CALCULATE # OF FRAMES THAT ANALYSIS TIME. IS BEHIND REAL TIME
TC-T∑ ->TC
I302 INFORMATIONAL MESSAGE
PRINT TC ON KEYBOARD
INDICATE WHICH WORD WAS SOUGHT AT TIME OF EXIT
PRINT WORD SYMBOL ON KB
PRINT # OF PATTERNS FOUND.
IF CONTROL ARRIVED HERE FROM RETURN, PROSODIC TIMING TESTS WERE NOT PERFORMED.
ARRIVE FROM RETURN?
OTHERWISE, PRINT LABEL OF LAST SUCCESSFUL PROSODIC TEST
PRINT SUMPAT∑
UNCIR1
JARCOF/2 ->DUMMY JARCOF-> DUMMY+2
SAVE WORD OFFSET TO DEST OF NEXT SUBJ T IN IDUM(1) SAVE BYTE OFFSET IN IDUM(2).
UNCDON 32->i 1->j 256->k
FILL AMPLITUDE ARRAY "JAMP" W/ AMP OF EACH FRAME. AMPLITUDE IS 32ND WORD OF EACH 32 WORD FRAME IN JIN.
AMPMOV
JIN_i ->JAMP_j (i+32)->i (j+1)->j (k-1)->k
SAVE VALUE, POINT TO AMP OF NEXT FRAME IN JIN.
k
> 0
< OR=0
LOOP UNITL AMPLITUDES OF 256 FRAMES SAVED.
UNSAVE RETURN ADDRESS.
RETURN
THIS IS THE ONLY EXIT POINT FOR THE SPIN4M REAL TIME WORD SPOT-SUBROUTINE.

SPIN3M SUBROUTINES

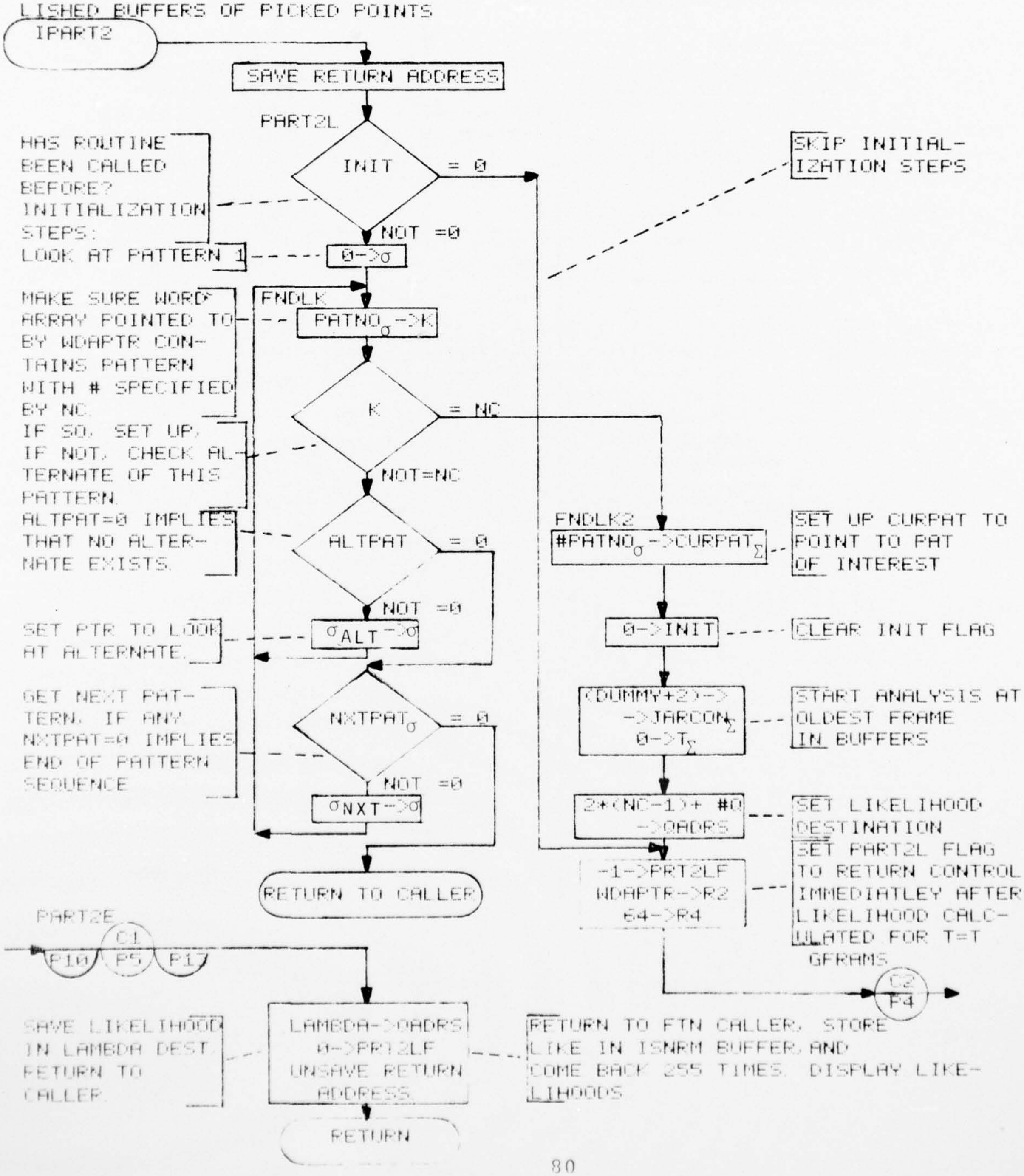
PAL NON-REAL TIME WORD DETECTION SUBROUTINE. CALLED W/ 2(R5)=IWRDNO

CALLABLE AFTER GO COMMAND, WITH JIN AND JARC CONTAINING SPECTRAL AND TIMING DATA FOR LAST 2.56 SECONDS. CALL PASSES # OF IWDAN ELEMENT THAT IS PTR TO WD DESCRIPTOR OF WD TO BE SOUGHT. USES PREVIOUSLY ESTABLISHED BUFFERS OF PICKED POINTS.



SPIN3M SUBROUTINES

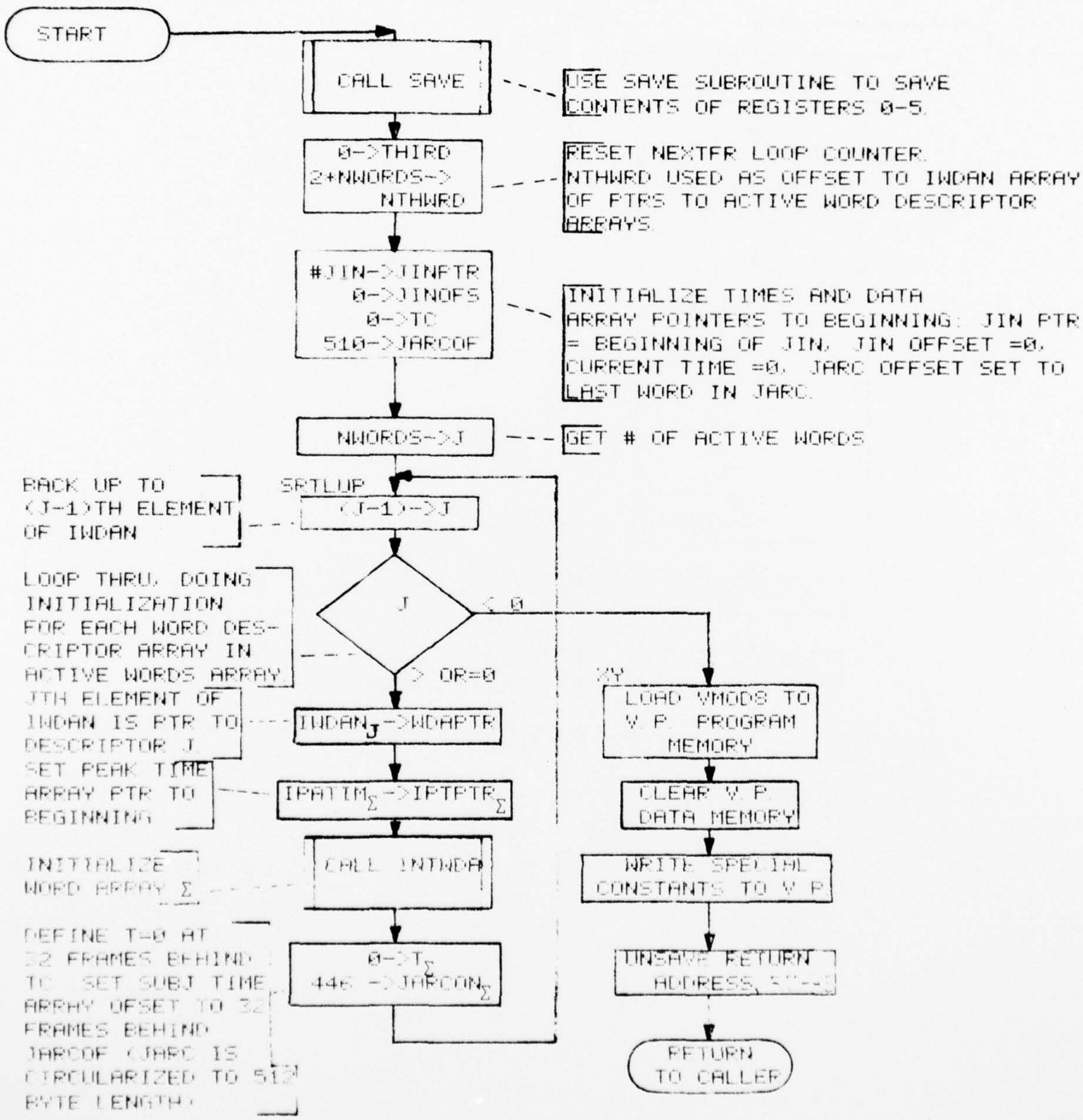
PAL NON REAL TIME LIKELIHOOD CALCULATION SUBROUTINE. CALLED WITH 2(R5)=INIT. CALLABLE AFTER GO COMMAND, WITH JIN AND JARC CONTAINING SPECTRAL AND TIMING DATA FOR LAST 2.56 SEC. CALLED WITH WDAPTR SET TO LAST WORD SOUGHT, AND NC=# OF PATTERN IN THAT WORD WHOSE LIKELIHOOD IS TO BE CALCULATED. USES PREVIOUSLY ESTABLISHED BUFFERS OF PICKED POINTS



SPIN3M SUBROUTINES

PAL SUBROUTINE CALLED BY SPIN4M, WITH NO ARGUMENTS PASSED.

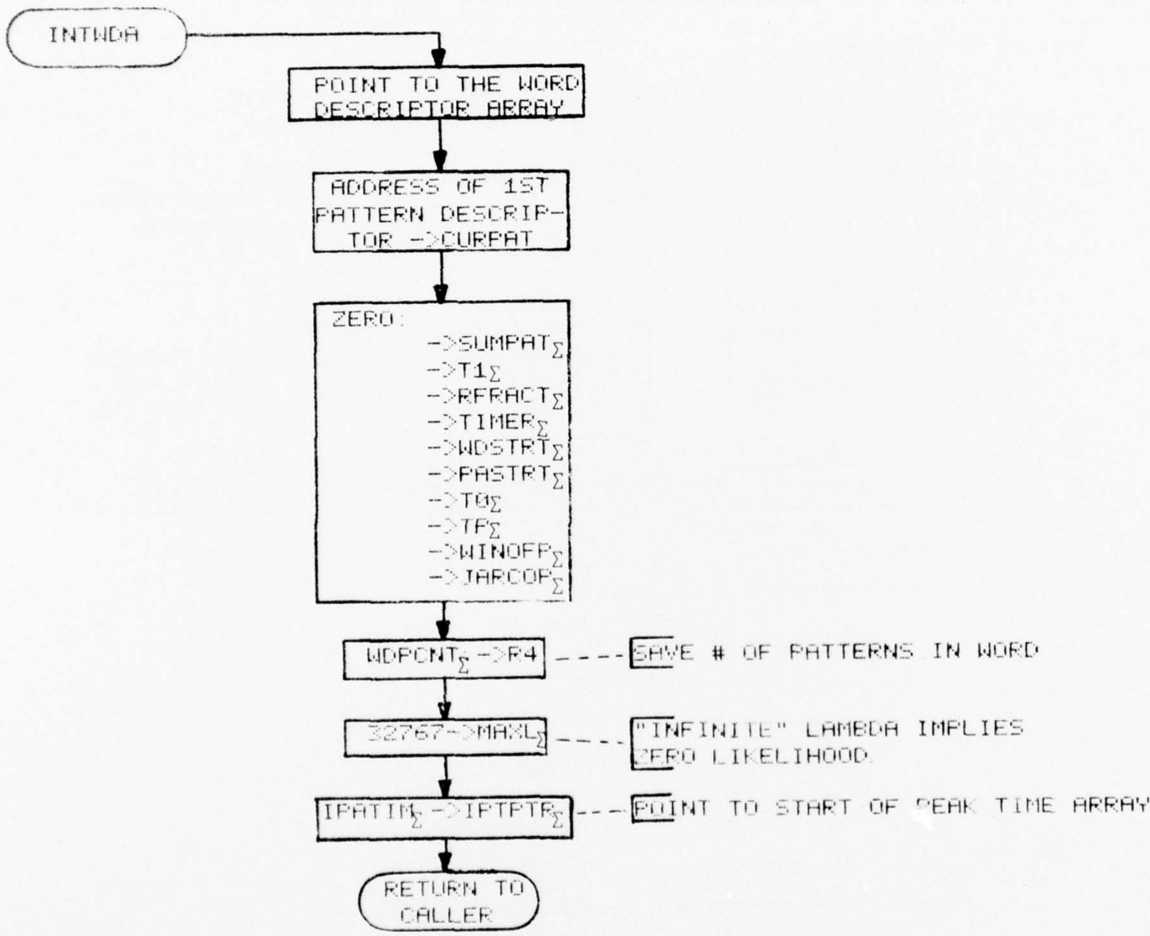
THIS SET-UP SUBROUTINE REQUIRES THAT ACTIVE WORDS TABLE "IWDAN" BE ESTABLISHED. USES THIS ARRAY OF ACTIVE WORDS ARRAY POINTERS PREVIOUSLY ESTABLISHED BY SETPTR SUBROUTINE.



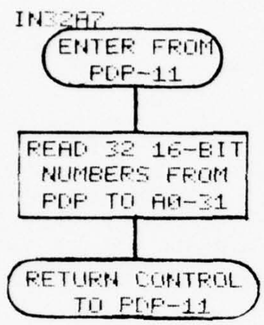
SPIN3M SUBROUTINES

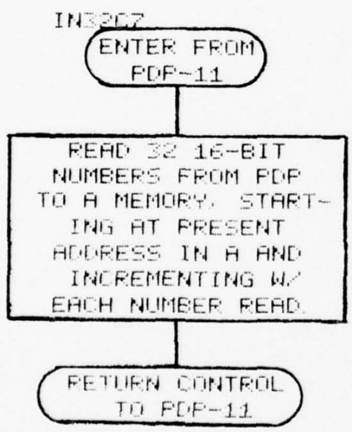
PAL WORD DESCRIPTOR INITIALIZATION SUBROUTINE.

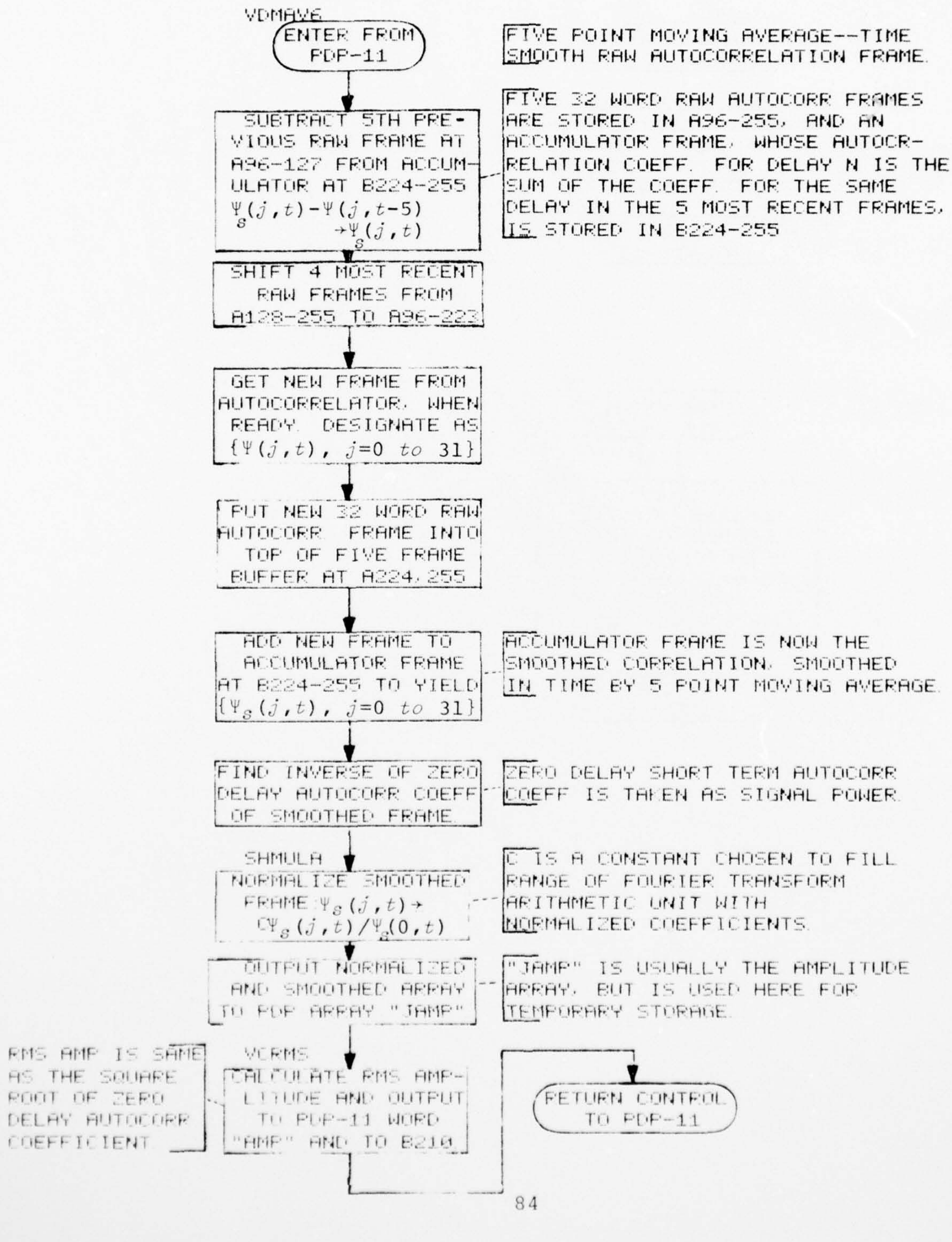
PRIOR TO CALLING INTWDA, WDAPTR MUST POINT TO WORDS ARRAY TO BE INITIALIZED. CALLED FROM PAL ONLY



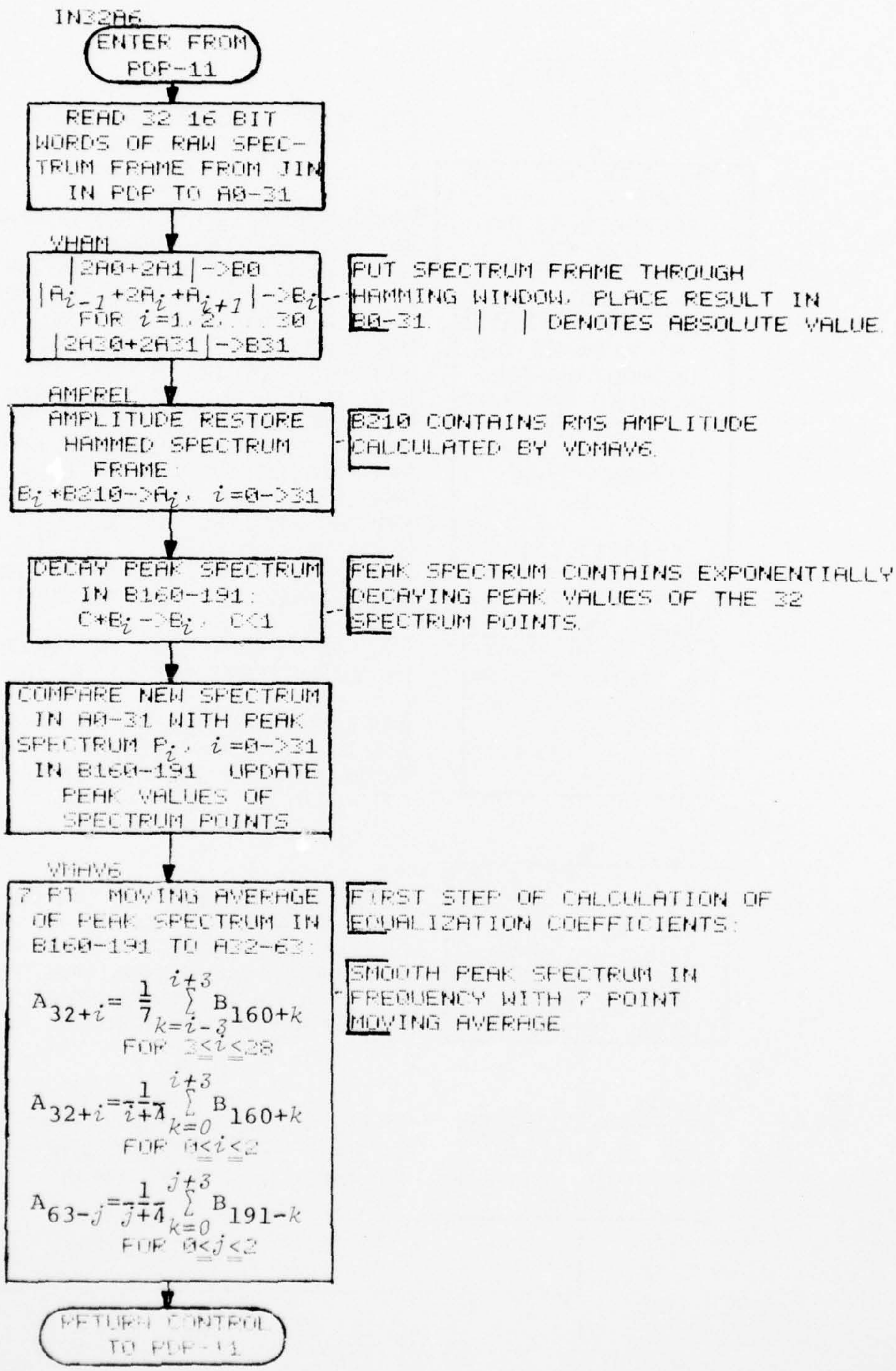
PIN3M V. P. ROUTINES

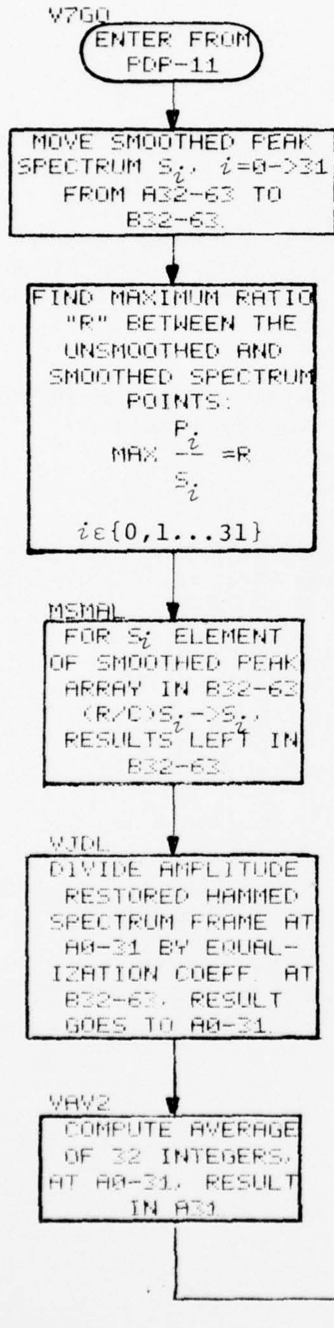





VDMAVE
ENTER FROM PDP-11
FIVE POINT MOVING AVERAGE--TIME SMOOTH RAW AUTOCORRELATION FRAME.
SUBTRACT 5TH PREVIOUS RAW FRAME AT A96-127 FROM ACCUMULATOR AT B224-255 $\Psi_s(j,t)-\Psi(j,t-5) \rightarrow \Psi_s(j,t)$
FIVE 32 WORD RAW AUTOCORR FRAMES ARE STORED IN A96-255, AND AN ACCUMULATOR FRAME, WHOSE AUTOCRRELATION COEFF. FOR DELAY N IS THE SUM OF THE COEFF. FOR THE SAME DELAY IN THE 5 MOST RECENT FRAMES, IS STORED IN B224-255
SHIFT 4 MOST RECENT RAW FRAMES FROM A128-255 TO A96-223
GET NEW FRAME FROM AUTOCORRELATOR, WHEN READY. DESIGNATE AS $\{\Psi(j,t),\ j=0\ to\ 31\}$
PUT NEW 32 WORD RAW AUTOCORR. FRAME INTO TOP OF FIVE FRAME BUFFER AT A224,255
ADD NEW FRAME TO ACCUMULATOR FRAME AT B224-255 TO YIELD $\{\Psi_s(j,t),\ j=0\ to\ 31\}$
ACCUMULATOR FRAME IS NOW THE SMOOTHED CORRELATION, SMOOTHED IN TIME BY 5 POINT MOVING AVERAGE.
FIND INVERSE OF ZERO DELAY AUTOCORR COEFF OF SMOOTHED FRAME.
ZERO DELAY SHORT TERM AUTOCORR COEFF IS TAKEN AS SIGNAL POWER.
SHMULA
NORMALIZE SMOOTHED FRAME: $\Psi_s(j,t) \rightarrow C\Psi_s(j,t)/\Psi_s(0,t)$
C IS A CONSTANT CHOSEN TO FILL RANGE OF FOURIER TRANSFORM ARITHMETIC UNIT WITH NORMALIZED COEFFICIENTS.
OUTPUT NORMALIZED AND SMOOTHED ARRAY TO PDP ARRAY "JAMP"
"JAMP" IS USUALLY THE AMPLITUDE ARRAY, BUT IS USED HERE FOR TEMPORARY STORAGE.
RMS AMP IS SAME AS THE SQUARE ROOT OF ZERO DELAY AUTOCORR COEFFICIENT
VCRMS
CALCULATE RMS AMPLITUDE AND OUTPUT TO PDP-11 WORD "AMP" AND TO B210
RETURN CONTROL TO PDP-11

SPIN3M V.P. ROUTINES





2ND STEP OF CALCULATION OF EQUALIZATION COEFFICIENTS:

THE OBJECT IS TO FIND A SCALE FACTOR, R, SUCH THAT
R*MAX(SMOOTHED PEAK ARRAY)
=MAX(UNSMOOTHED PEAK ARRAY)
THIS MAXIMUM RATION IS CLEARLY THE MAXIMUM RATIO OF ANY POINT IN THE CURRENT AMPLITUDE RESTORED HAMMED SPECTRUM FRAME TO THE CORRESPONDING SMOOTHED PK ARRAY POINT. THUS IF WE SCALE THE SMOOTHED PEAK ARRAY BY A MULTIPLICATIVE FACTOR OF R/C, ALL VALUES OF THE EQUALIZED FRAME, I.E., HAMMED SPECTRUM DIVIDED BY SCALED SMOOTHED PK ARRAY, WILL LIE BETWEEN 0 AND C. THIS ALLOWS UTILIZATION OF FULL RANGE OF THE ARITHMETIC UNIT.

3RD AND FINAL STEP OF CALCULATON OF EQUALIZATION COEFFICIENTS:

MULTIPLY SMOOTHED PEAK ARRAY BY RATIO R/C TO YIELD 32 EQUALIZATION COEFFICIENTS.

EQUALIZE CURRENT AMPLITUDE RESTORED HAMMED SPECTRUM FRAME TO YIELD "EQUALIZED FRAME." EACH POINT OF AMP RESTORED HAMMED SPECTRUM FRAME IS DIVIDED BY CORRESPONDING COEFF IN EQUALIZATION ARRAY.

FIND THE AMPLITUDE OF THE EQUALIZED SIGNAL, AND REPLACE 32ND SPECTRUM (4 KHZ) WITH THIS VALUE.

C1
P21

IWT2

FORM ARRAY B192-207 OF WEIGHTED DISTANCES ALONG 1ST 16 SPECTRAL AXES IN MULTIDIMENSIONAL SPACE OF FREQUENCY COMPONENTS:
FOR i=0 TO 15,
$[A(0+i)-B(192+i)] * IWT_i \rightarrow B(192+i)$

1ST 15 POINTS OF PREVIOUS EQUALIZED FRAME ARE SAVED IN B192-207. LATEST EQUALIZED FRAME STORED AT A0-31. WEIGHTING CONSTANTS STORED IN PDP ARRAY IWT. ONLY 1ST 16 SPECTRUM POINTS ARE USED TO CALCULATE ARC LENGTH (INTERFRAME DISTANCE OR "SUBJECTIVE TIME").

INTL2

ACCUMULATE SUM OF 16 WEIGHTED AXIAL DISTANCES AND STORE IN B221.

METRIC FOR 16 DIMENSIONAL SPECTRUM SPACE IS WEIGHTED "TAXICAB METRIC":

$$D[(X_1, X_2, \ldots X_n), (Y_1, Y_2, \ldots Y_n)] = \sum^{n} C_i |X_i - Y_i|,$$

WHERE THE C_i ARE WEIGHTS. SUM OF WEIGHTED AXIAL DISPLACEMENTS BETWEEN COORDINATES OF TWO FRAMES IS CALLED "SUBJECTIVE TIME" ELAPSED BETWEEN THE FRAMES.

ATOB3

SAVE 1ST 16 POINTS OF NEW EQUALIZED SPECTRUM AT A0-31 IN SPACE FOR PREVIOUS EQUALIZED SPECTRUM AT B192-207.

DIVIDE EQUALIZED FRAME AVERAGE VALUE IN A31 BY 2 TO GIVE AMPLITUDE IN B0.

LOPOUT

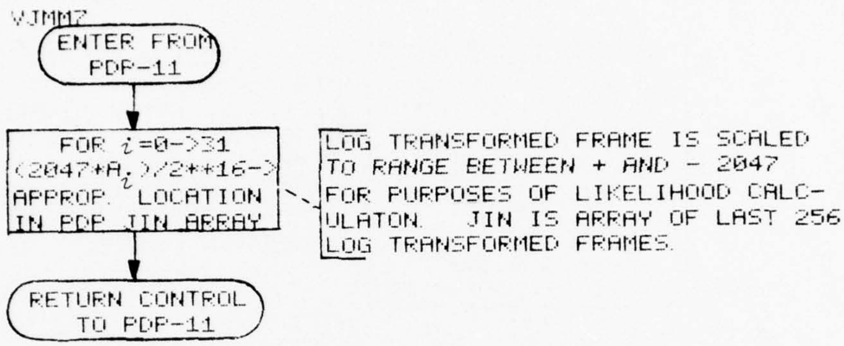
WRITE OUT ARC LENGTH INCREMENT (SUBJECTIVE TIME) IN B221 TO PDP WORD "ARC"

LOG TRANSFORMATION OF A0-31 WITH RESPECT TO AVERAGE B0. RESULT GOES IN A0-31
FOR i=0->31,
$$\frac{(A_i - B0)*C}{(A_i + B0)} \rightarrow A_i$$

PERFORM QUASI-LOG TRANSFORMATION ON EQUALIZED FRAME, TO PRODUCE SOFT THRESHOLD AND SATURATION EFFECT FOR SPECTRAL INTENSITIES DEVIATING GREATLY FROM AVERAGE. B0 CONTAINS AVERAGE VALUE OF EQUALIZED FRAME. SATURATES TO +C FOR A INFINITELY LARGE, AND -C FOR A_i=0. C=2**16

RETURN CONTROL TO PDP-11

SPIN3M V.P. ROUTINES



VCLK7

ENTER FROM PDP-11

COMPUTE THE DIFFERENCES BETWEEN TEST PATTERN VALUES x_i IN A0-95 AND REFERENCE PATTERN MEAN VALUES μ_i IN PDP-11 STATISTICS BUFFER.
FOR i=0->95,
$A_i - \mu_i \rightarrow B_i$

VCLK7 ENTERED WITH V.P. BUS ADDRESS REGISTER POINTING TO REFERENCE PATTERN MEAN VALUES.
THESE STEPS ARE TO CALC. LIKELIHOOD THAT THE THREE FRAMES IN A MEMORY ARE SAME PATTERN AS ONE CURRENTLY SOUGHT. FOR A SOUGHT PATTERN WHOSE REFERENCE PAT. MEAN VALUES ARE μ_i, i=0->95, AND STANDARD DEV. VALUES ARE σ_i i=0->95, LIKELIHOOD P THAT PATTERN WITH VALUES x_i IS SAME AS REFERENCE PATTERN IS GIVEN BY

$$P = K\left[\left[\sum_{i=0}^{95}\left((x_i-\mu_i)\frac{1}{\sigma_i}\right)^2\right]+L\right]$$

WHERE K IS A SCALE FACTOR AND L IS A LOG OFFSET TERM. P IS INVERSELY RELATED TO CLOSENESS OF FIT OF TEST PATTERN TO REFERENCE PATTERN.

RETURN CONTROL TO PDP-11

VCLK27

ENTER FROM PDP-11

VCLK27 ENTERED W/ V.P. BUS ADDRESS REGISTER POINTING TO MULTIPLICATIVE INVERSES OF REFERENCE PATTERN STANDARD DEVIATIONS.

MULTIPLY DIFFERENCES IN B0-95 BY INVERSES OF STANDARD DEVIATIONS AND SQUARE. ACCUMULATE SQUARES IN V.P. "F" REGISTER. FOR i=0->95,

$$F+\left[\frac{B_i}{\sigma_i}\right]^2 \rightarrow F$$

F->B0

ADD LOG OFFSET TERM FROM PDP STATISTICS TO ACCUMULATED SQUARES IN B0
B0+L->F

IN STATISTICS BUFFER, FOLLOWING STANDARD DEVIATIONS, IS A LOG OFFSET TERM GENERATED BY THE SUM OF THE LOGS OF STANDARD DEVIATIONS FOR POINTS IN THIS 96 POINT PATTERN.

SCALE LIKELIHOOD IN F BY CONSTANT K
K*F->F

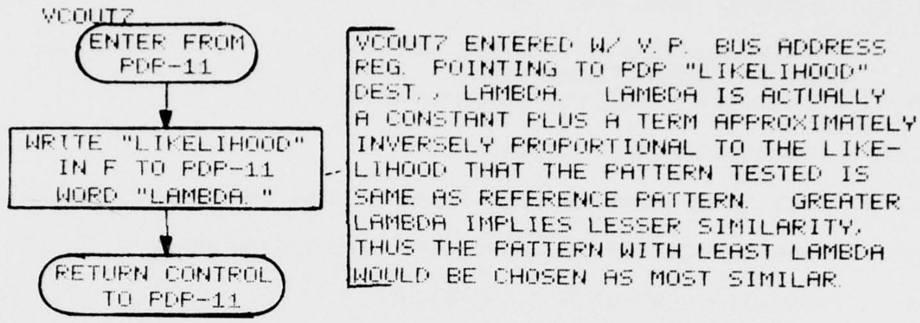
RETURN CONTROL TO PDP-11

AD-A044 605 DIALOG SYSTEMS INC CAMBRIDGE MA F/G 17/2
KEY WORD CLASSIFICATION.(U)
AUG 77 S L MOSHIER, P N LEIBY, R E SMITH F30602-75-C-0171
UNCLASSIFIED RADC-TR-77-122 NL

2 OF 2
ADA044605

END
DATE
FILMED
10-77
DDC

SPIN3M V.P. ROUTINES



DICTIONARY OF SYMBOLS AND VARIABLES FOUND IN FLOWCHART

#	NUMBER, INDICATES "ADDRESS OF" WHEN PRECEEDING A VARIABLE.
A	A MEMORY, ONE OF TWO 256 WORD X 16 BIT DATA MEMORIES IN THE VECTOR COMPUTER.
A0	1ST WORD OF VECTOR COMPUTER A MEMORY.
ADDR	ADDRESS (ABBREVIATION)
ALTPAT	PATTERN PARAMETER LIST ELEMENT: PTR TO CURRENT PAT. ALTERNATIVE.
AMP	DESTINATION OF RMS AMPLITUDE CALCULATED BY V.P..
AN	(N-1)TH WORD OF V.C. A MEMORY.
ARC	REPOSITORY IN PDP-11 FOR SUBJECTIVE TIME OF CURRENT INPUT FRAME AS CALCULATED BY THE V.P..
AUTOCOR	AUTOCORRELATOR (ABBREVIATION).
B	B MEMORY, ONE OF TWO 256 WORD X 16 BIT DATA MEMORIES IN THE VECTOR COMPUTER.
B0	1ST WORD OF VECTOR COMPUTER B MEMORY.
BN	(N-1)TH WORD OF VECTOR COMPUTER B MEMORY
COEFF	COEFFICIENT (ABBREVIATION)
COMP.	COMPUTE (ABBBREVIATION)
CURPAT	WDA ELEMENT: POINTER TO PATTERN PARAMETER LIST FOR CURRENT PATTERN SOUGHT.
DUMMY	ARRAY USED FOR PASSAGE OF VARIABLES TO FORTRAN. AT TERMINATION OF REAL TIME SEARCH IT CONTAINS OFFSET TO OLDEST JARC TIME.
F LATCH	VECTOR PROCESSOR STORAGE REGISTER.
FRAM1	BUFFER OF POINTERS TO 1ST FRAME PICKED FOR PATTERN CORRESPONDING TO EACH FRAME IN JIN. SAME CONSTRUCTION AS JARC.
FRAM2	BUFFER OF POINTERS TO 2ND FRAME PICKED FOR PATTERN CORRESPONDING TO EACH FRAME IN JIN. SAME CONSTRUCTION AS JARC.
FRAM3	BUFFER OF POINTERS TO 3RD FRAME PICKED FOR PATTERN CORRESPONDING TO EACH FRAME IN JIN. SAME CONSTRUCTION AS JARC.
FRAME	BASIC UNIT OF INPUT DATA: ORIGINALLY 32 AUTOCORRELATION COEFFICIENTS, SUBSEQUENTLY PREPROCESSED INTO A 32 POINT SPECTRUM. ONE FRAME IS A UNIT OF TIME EQUAL TO 10 MILLISECONDS BY THE CLOCK.
FTN	FORTRAN (ABBREVIATION)
I302	INFORMATIONAL MESSAGE PRINTED ON KB AFTER TERMINATION OF REAL TIME SEARCH, ACCOMPANIED BY VALUE OF TC-T.
I303	INFORMATIONAL MESSAGE PRINTED ON KB IF ANALYSIS TIME FALLS TOO FAR BEHIND REAL TIME, ACCOMPANIED BY TC-T.
IDT	REAL TIME SEPARATION BETWEEN PICKED FRAMES OF PATTERN, GIVEN IN UNITS OF 10 MS.
IFILT	ARRAY OF COSINE TRANSFORM COEFFICIENTS USED TO CALCULATE SPECTRUM OF INPUT FRAME.
INIT	INITIALIZATION FLAG SET BY FORTRAN BEFORE CALLING IPART2, AND CLEARED BY IPART2 AFTER INITIALIZATION.
INTWDA	SUBROUTINE TO INITIALIZE THE WORD DESCRIPTOR ARRAY CURRENTLY POINTED TO BY WDAPTR.
IPART2	NON-REAL TIME LIKELIHOOD CALCULATION SUBROUTINE. CALCS LIKE

	THAT PATTERN ASSOC. WITH EACH FRAME IS SOUGHT PATTERN.
IPATIM	WDA ELEMENT: POINTER TO START OF PEAK TIME ARRAY.
IPSTAR	WORKSPACE FOR CALCULATION OF DETECTED WORD'S PROSODIC TIMING PARAMETERS.
IPTPTR	WDA ELEMENT: PTR TO DESTINATION OF NEXT PEAK TIME IN PEAK TIME ARRAY (INITIALLY = IPATIM).
ISNRM	BUFFER USED BY NON-REAL TIME ROUTINES TO STORE EITHER LIKE. OF EACH FRAME (IPART2), OR # OF PATS FOUND AT EACH FRAME (SPN4NR).
IWDAN	ARRAY OF POINTERS TO TARGET WORD WORD DESCRIPTOR ARRAYS
IWDSYM	ARRAY OF SYMBOLS ASSOCIATED WITH EACH TARGET WORD, IN ORDER OF TARGET WORD SPECIFICATION.
IWT	ARRAY OF SUBJECTIVE TIME WEIGHTS. SUBJ TIME IS BASED OF THE SUM OF WEIGHTED SPECTRAL CHANGES.
JAMP	ARRAY OF AMPLITUDES OF EACH FRAME IN JIN, FILLED ONLY AFTER TERMINATION OF REAL TIME SEARCH.
JARC	256 WORD ARRAY OF 16 BIT FRAME SUBJECTIVE TIMES (CIRCULAR).
JARCEN	LAST WORD OR END OF JARC.
JARCOF	OFFSET TO JARC INDICATING DESTINATION OF NEXT INPUT FRAME'S SUBJECTIVE TIME.
JARCON	WDA ELEMENT: CONTAINS BYTE OFFSET IN JARC TO LOCATION OF SUBJECTIVE TIME CORRESPONDING TO THIS WORD'S ANALYSIS TIME.
JARCOP	WDA ELEMENT: JARC OFFSET TO SUBJ. TIME CORRESPONDING TO PEAK LIKELIHOOD.
JIN	8K WORDS ARRAY OF 256 32 WORD SPECTRUM FRAMES (CIRCULAR)
JINOFS	BYTE OFFSET TO JIN GIVING DESTINATION OF NEXT 32 POINT SPECTRUM FRAME.
JINPTR	POINTER TO DESTINATION OF NEXT INPUT SPECTRUM FRAME IN JIN. JINPTR = #JIN + JINOFS
KB	KEYBOARD (ABBREVIATION)
LAMBDA	PATTERN SIMILARITY MEASURE (APPROX. INVERSELY PROPORTIONAL TO LIKE. THAT TEST PATTERN IS SAME AS THE REFERENCE PATTERN).
LC	ALPHABETIC ARGUMENT OF KEYBOARD COMMAND.
LIKE.	LIKELIHOOD (ABBBREVIATION)
MAXL	WDA ELEMENT: PEAK LIKELIHOOD FOUND FOR CURPAT SO FAR.
MEANS	PATTERN PARAMETER LIST ELEMENT: POINTER TO START OF MEAN VALUE STATISTICS FOR THIS PATTERN.
MOD	MODULO: X MODULO N = REMAINDER OF X/N
MVALUE	MEAN VALUE OF DETECTED WORD'S PATTERN PEAK TIMES.
NC	NUMERICAL ARGUMENT OF KEYBOARD COMMAND.
NRFLG	NON-REAL TIME WORD DETECTION FLAG, SET IF SPN4NR EXECUTING.
NTHWRD	BYTE OFFSET IN IWDAN TO PTR POINTING TO WORDS ARRAY OF WORD CURRENTLY SOUGHT. BYTE OFFSET TO SYMBOL IN IWDSYM.
NWORDS	NUMBER OF WORDS TO BE SOUGHT (AT PRESENT A MAXIMUM OF 2).
NXTPAT	PATTERN PARAMETER LIST ELEMENT: PTR TO PAT. SUCCEEDING CURPAT.
PAL	PDP-11 ASSEMBLY LANGUAGE.
PASTRT	WDA ELEMENT: FLAG SET IF CURRENT PATTERN SOUGHT HAS CROSSED LIKELIHOOD THRESHOLD, AND IS BEING TRACKED FOR PEAK.
PATTERN	BASIC SOUND UNIT, COMPOSED OF THREE SPECTRAL FRAMES EQUALLY SPACED IN SUBJECTIVE TIME.
PC	PROGRAM COUNTER, I.E., R7 FOR PDP-11

PDP — PROGRAM DATA PROCESSOR-- DEC MACHINE.

PK — PEAK (ABBREVIATION)

POINTER — A POINTER TO X IS A WORD CONTAINING THE ADDRESS OF X

PRT2LF — FLAG SET (NOT. = 0) IF NON-REAL TIME LIKELIHOOD ROUTINE RUNNING.

PTMAR — WDA ELEMENT: POINTER TO ARRAY OF PROSODIC TIMING PARAMETER MAXIMUM AND MINIMUM VALUES.

PTR — POINTER (ABBBREVIATION)

QADRS — DESTINATION OF LIKELIHOOD CALCULATED BY IPART2 FOR EACH FRAME, USED TO PASS LIKE. TO FORTRAN.

R0 — REGISTER 0

R1 — REGISTER 1

R2 — REGISTER 2

R3 — REGISTER 3

R4 — REGISTER 4

R5 — REGISTER 5

R6 — REGISTER 6

R7 — REGISTER 7

SAVE — SUBROUTINE TO SAVE CONTENTS OF REGISTERS 0 THRU 5.

SETPTR — SUBROUTINE CALLED BY FORTRAN TO SET TARGET WORDS AND SET POINTERS TO THEIR STATISTICS IN THEIR WORD DESCRIPTOR ARRAYS.

SP — STACK POINTER, I.E., R6 FOR PDP-11

SPIN3M — SPEECH INTERPRETER 3, MULTIPLE WORD SPOTTING (ENTIRE PROGRAM).

SPIN4M — SPEECH INTERPRETER 4, MULTIPLE WORD SPOTTING, REAL TIME PAL SUBROUTINE

SPN4NR — NON-REAL TIME WORD SPOTTING SUBROUTINE. SEEKS ONE TARGET WORD IN DATA SAVED IN BUFFERS FROM REAL TIME RUN.

START — BUFFER INITIALIZATION SUBROUTINE (DOES NOT CHANGE TARGET WORDS).

STDS — PATTERN PARAMETER LIST ELEMENT: POINTER TO START OF STANDARD DEVIATION STATISTICS FOR THIS PATTERN

SUBJ. — SUBJECTIVE (ABBBREVIATION)

SUMPAT — WDA ELEMENT: NUMBER OF PATTERNS IN WORD PATTERN SEQUENCE DETECTED SO FAR.

SWR — CONSOLE SWITCH REGISTER.

T — WDA ELEMENT: ANALYSIS TIME, TIME (IN REAL TIME UNITS) OF FRAME ACTUALLY BEING ANALYZED. TC-256 < T < TC

T0 — WDA ELEMENT, EXPIRATION TIME OF PEAK TRACKING FOR THIS PATTERN. T0 = (TP OF FIRST THRESH CROSSING) + TRKTIM

T1 — WDA ELEMENT: TP+WINDOW, EXPIRATION TIME OF SEARCH FOR CURRENT PATTERN SOUGHT.

TC — CURRENT (REAL) TIME, INCREMENTED 1 UNIT = 10 MS FOR EVERY NEW AUTOCORRELATION INPUT FRAME ABOVE AMPLITUDE THRESHHOLD.

THIRD — NEXTFR LOOP COUNTER, INCREMENTED WITH EACH LOOP FROM NEXTFR. AFTER THIRD LOOP, GO GET NEW INPUT FRAME.

THR — AMPLITUDE THRESHOLD, BELOW WHICH INPUT FRAME IS IGNORED

THRESH — PATTERN PARAMETER LIST ELEMENT: INDICATES LIKELIHOOD

	THRESHOLD FOR THAT PATTERN.
TIMER	WDA ELEMENT: (TP OF PAT 1)+WDLMIN = EARLIEST ACCEPTABLE TIME FOR TOTAL WORD END.
TOTIME	TOTAL REAL TIME DURATION OF DETECTED WORD.
TP	WDA ELEMENT: TIME OF CURPAT PEAK LIKELIHOOD
TRKTIM	LENGTH OF INTERVAL FOR WHICH EACH PATTERN LIKELIHOOD PEAK IS TRACKED.
UNSAVE	SUBROUTINE TO RESTORE PREVIOUSLY SAVED VALUES OF REGISTERS 0-5.
VBA	VECTOR COMPUTER BUS ADDRESS REGISTER
V. C.	VECTOR COMPUTER (ABBREVIATION)
V. P.	VECTOR PROCESSOR (ABBBREVIATION)
VPG	VECTOR COMPUTER PROGRAM COUNTER REGISTER
WD	WORD (ABBREVIATION)
WDA	WORD DESCRIPTOR ARRAY. ARRAY OF REFERENCE AND STATUS INFORMATION CONCERNING ONE OF THE TARGET WORDS. SEE DATA STRUCTURES SECTION.
WDAPTR	POINTER TO WORD DESCRIPTOR ARRAY OF WORD CURRENTLY SOUGHT.
WDLMIN	WDA ELEMENT: MINIMUM WORD LENGTH GIVEN IN # OF FRAMES.
WDPCNT	WDA ELEMENT: # OF PATTERN DETECTIONS COMPRISING A WORD DETECTION.
WDSTRT	WDA ELEMENT: FLAG SET IF FIRST PATTERN HAS BEEN FOUND FOR WORD SOUGHT. "WORD STARTED".
WINDOW	PATTERN PARAMETER LIST ELEMENT: # OF FRAMES TO END OF DETECTION WINDOW FOR NEXT PATTERN.
W/	WITH (ABBREVIATION)

III DIALOG VECTOR PROCESSOR

Introduction

The dialog vector computer is a high-speed single cycle processor designed for rapid vector arithmetic. The vector computer consists of several digital arithmetic units and data memories connected to each other via three 32-bit data buses. The PDP-11 host computer exerts primary control over the computer via four device registers on the PDP-11 unibus. The first of these is the vector computer "PC" register which is used to specify the starting address of the vector computer program. Modification of this register places the vector computer in "run" state. The second register is the unibus address register which the vector computer uses when transferring data to or from the host computer's memory. The third register is the program "LOAD" register. This register is used to load the vector computer program memory from the host computer. This operation is the only way to place data in the vector computer program memory. The fourth register is used for hardware testing.

Vector Computer Unibus Address:

PC register	-	167730
Unibus address register	-	167732
Program load register	-	167734
Diagnostic register	-	167736

The cross-assembler named XASM is used to create binary program images suitable for loading in the vector computer via the program load register. XASM takes source text using syntax similar to the PAL-11 assembler and outputs an object module compatible with the system linker. The object module is given a global name so that the programmer may load the module with a single "MOV" instruction.

```
.GLOBL NAME
MOV  #NAME,@#167734  ;loads module called "NAME"
```

In essence, XASM simply provides symbolic names for bit patterns representing XASM instructions and a syntax for specifying bit settings within the vector computer instruction. XASM permits the programmer to specify instruction settings as octal, decimal, or binary numbers, or by using built-in or user-defined symbols. XASM has a large set of standard symbolic names for computer devices and operations.

Vector Processor Instruction Classes

The vector processor recognizes four different operation codes. They are:

```
00 - Arithmetic-logic instructions   (ALU)
01 - Data class instructions         (MISCELLANEOUS)
10 - Bus transfer instructions       (BUS)
11 - Program branch instructions     (BC)
```

Each instruction takes one machine cycle (120 nsec) to execute. Each instruction has four common bits in addition to the op-code. Three of these bits are used to enable the three data buses for transfers during the instruction execution (if desired). The fourth bit is the "REPEAT" bit. If set, the instruction will be repeated for the number of times stored in the "REPEAT COUNTER". These bits may be set by using the ".COM" instruction. The rest of the 32-bit instruction is used to specify the exact function. Arithmetic (ALU) instructions cause the ALU to perform some single-cycle arithmetic functions using the contents of the ALU registers (The "AR", "BR", and "FR"). The bus transfer instructions are used to set up the three processor buses ("A", "B", and "D") to transfer between specified processor registers or devices. A "BUS" instruction will actually perform the transfer if the "BUS ENABLE" common bits are set for the buses involved. The branch instructions are used to conditionally branch in a vector computer program and optionally save a return address. The branch address may be part of the instruction, the previously saved return address, or may come from the processor D-bus. The data class instructions are generally processor control instructions.

Data bus structure; the BUS class instruction

There are three principal data buses, named A, B, and D. Each bus is 32 bits wide. During an instruction cycle, which lasts 120 nanoseconds, there may be at most one source and one destination for data on each bus. Each source and each destination has a four-bit address code which is set up by executing a BUS class instruction. The address code is transmitted on an independent address bus. For example, the "B memory" scratchpad is a possible source for the B bus and the "B register" input to the arithmetic unit is a possible destination for the B bus. The BUS instruction has a total of six address fields, which must be specified in a particular order when writing a BUS instruction in the assembly language. The protocol is

```
BUS  A bus source, A bus destination,
     B bus source, B bus destination,
     D bus source, D bus destination
```

Each data source or destination has a mnemonic assembly language code which may be entered at the appropriate spot in a BUS class assembly language instruction. The mnemonics appear as port labels in Figure 11.

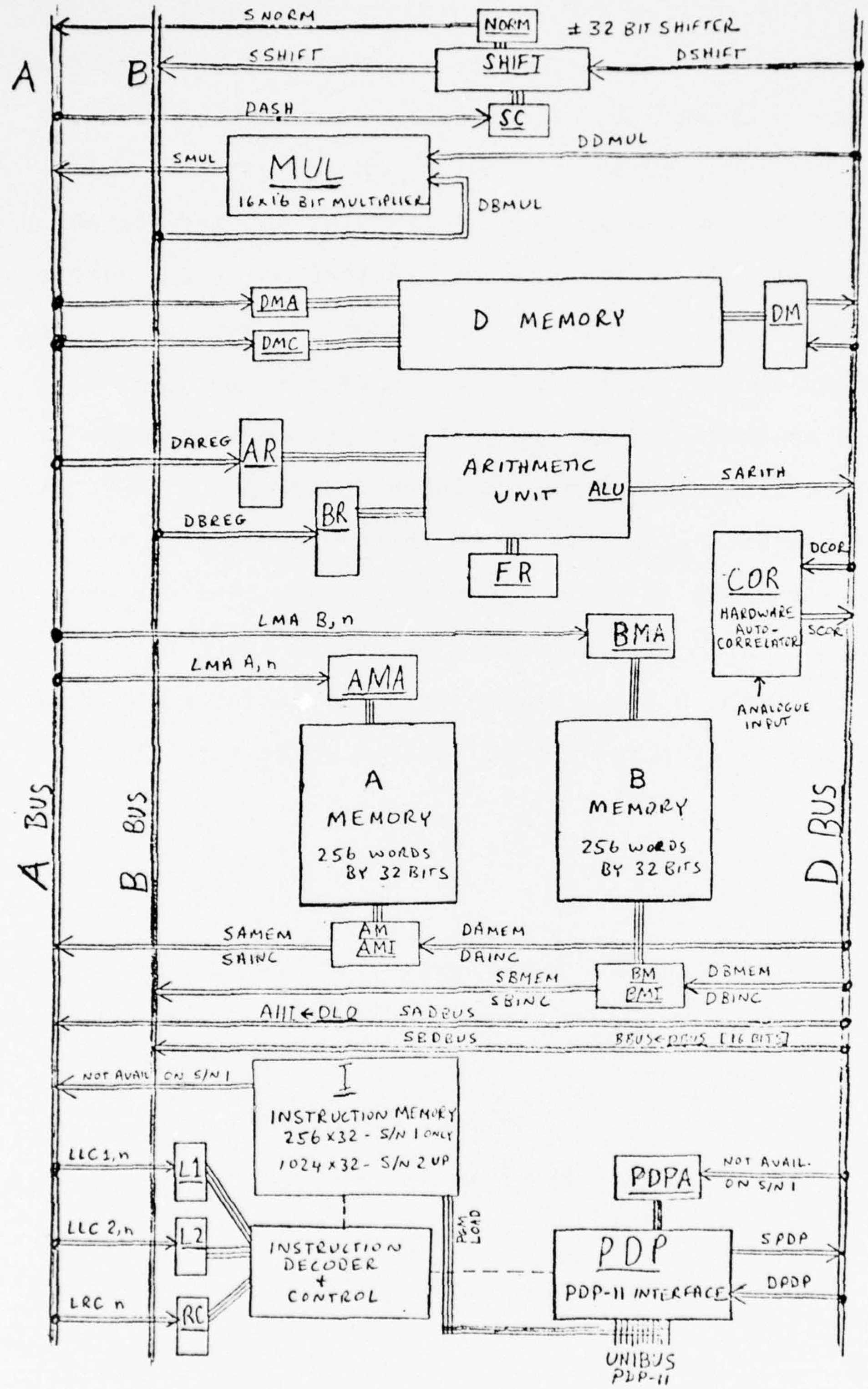


Figure 11. Dialog Vector Processor

The sources and destinations stipulated in a BUS instruction remain active until superseded by subsequently executing another BUS instruction. However, no data is actually transmitted or received on any data bus unless the bus has been enabled explicitly by setting its associated bus enable bit in the program instruction word. With one exception explained in the next section, any bus may be enabled on any instruction, and data transfer will take place as specified in the most recently executed BUS instruction. If a bus is enabled during a BUS instruction, data transfer will take place in accordance with the specifications in that same instruction. The bus enable bits are bit 3 for the A bus, bit 4 for the B bus, and bit 5 for the D bus. In the assembly language any desired combination of buses may be enabled by writing

.COM Arg_1, Arg_2, ...

on the line immediately following the instruction on which the bus or buses are to be enabled. The assembly language arguments Arg_i are

AE to enable the A bus

BE to enable the B bus

DE to enable the D bus.

The only other legal argument to .COM is RE, which is explained in the section on the do-loop and repeat counters. Table IV gives a complete listing of legal bus source and destination mnemonics.

The following characteristics of the processor hardware must be understood for proper programming results. All operations in the processor are driven by a master clock whose period is 0.12 microsecond. All bus sources are driven by tri-state (high, low, off) drivers, and all destinations receive data via clocked latches. Thus at the beginning of each master clock cycle, data is gated onto all enabled buses, but no data is received at any bus destination until the end of that cycle (i.e., the beginning of the next cycle). All devices (e.g., memories, arithmetic unit, ...) connected to the buses have characteristic propagation delay times which must be honored to ensure that their outputs are valid at the time they are actually received at the desired destination. For example, in most operations the arithmetic unit requires one clock cycle for its outputs to settle. Thus to add X and Y, the programmer may cause X to be written into the A register and Y to be written into the B register on completion of instruction n, but the sum X+Y will not be valid at any enabled destination until one clock cycle later, at the end of the next following instruction n+1. In array

operations this means that fastest execution speed is obtained by filling a short "pipeline" consisting of a chain of registers before entering a repetitive software loop to step through the array.

Because the processor is designed especially for fast execution of array operations, each data memory is accessed via its own address counter which may be set to increment automatically whenever the memory is referenced on an enabled bus. The address counter is treated as a separate device by the BUS instruction. For example, to access address a in the B memory, one first deposits the number a in the B memory address counter, which is a destination on the A bus. The contents of B memory location a will become valid and available for transfer by the end of the next following instruction. Alternatively, data may be *written* into the B memory at location a on the same instruction which loaded the address counter; however the actual write cycle occurs while the next following instruction is being executed, so it is not possible to read from the memory until the second instruction following a write instruction. A succession of read operations or write operations may be executed on contiguous instructions, but the A and B scratchpad memories must be given one bus cycle to recover when switching from writing to reading; this recovery cycle may be used to load a new address into the address counter,

without disturbing the write operation, since the address information is not actually changed until the end of the instruction.

TABLE IV. BUS BUS CLASS INSTRUCTION CODES

FORMAT:

BUS: A-Source, A-Destination, B-Source, B-Destination
D-Source, D-Destination

This instruction sets all bus addresses for all processor buses.

A destination uses .word Ø,↑ B0000111100000000 as a mask with these operands:

AR	- A-register in ALU	17
RC	- Repeat counter	16
SC	- Shift counter	15
L1	- Loop counter 1	12
L2	- Loop counter 2	14
DMC	- D-memory control register	10
DMA	- D-memory address register	7
BMA	- B-memory address register	6
AMA	- A-memory address counter	5

A- Source uses .word Ø, ↑B1111000000000000 as a mask with these operands:

AM	- A-memory output	17
AMI	- A-memory output w/auto increment	16
DBUS	- Low 16 bits of D bus connected to high 16 bits of A bus	15
NORM	- Normalizer output	14
MUL	- Multiplier output	13
I	- data from instruction	1

B- Source uses .word ↑B0000000000001111,0 as a mask with these operands:

BM	- B-memory output	17
BMI	- B-memory output, auto incremented	16
SH	- Shift output	15
DBUS	- Low 16 bits of D bus connected to low 16 bits of B bus	14

B- Destination uses .word ↑B0000000011110000,0 as a mask with operands:

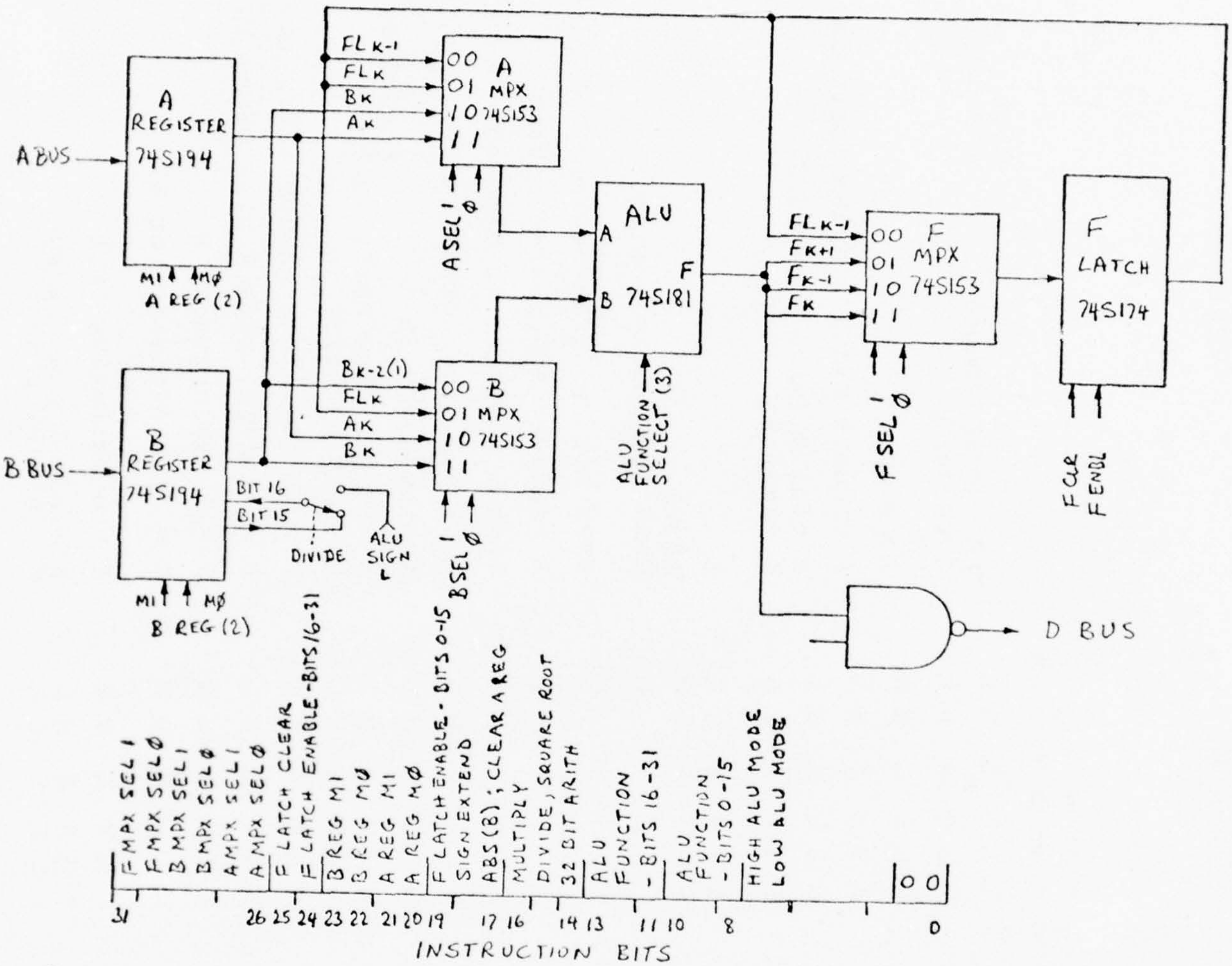
BR	- B-register in ALU	17
MUL	- Multiplier input	16

D- Source uses .word ↑B0000111100000000,0 as a mask with these operands:

PDP	- PDP-11 data bus latch	17
ALU	- output of ALU	16
COR	- Auto correlator output	14
DM	- D-memory output	6

D- Destination uses .word ↑B1111000000000000,0 as a mask with these operands:

PDP	- PDP-11 data bus latch	17
BM	- B-memory input	16
AM	- A-memory input	15
BMI	- B-memory auto increment	14
AMI	- A-memory auto increment	13
SH	- Shifter input	12
MUL	- Multiplier input	11
COR	- Auto correlator input	10
PDPA	- PDP bus address register	7
DM	- D-memory input	6



NOTES:

(1) B SEL = 11 $\Rightarrow \begin{cases} B_{K-2} \rightarrow B_K, & K \in [16,31] \\ FL_{K+16} \rightarrow B_K, & K \in [0,15] \end{cases}$

(2) A + B REGISTER CODES:

0 0	HOLD
0 1	SHIFT UP
1 0	SHIFT DOWN
1 1	LOAD

FIGURE 12.

ARITHMETIC INSTRUCTION

(3) ALU FUNCTIONS:

0 0 0	NOT A	1 0 0	ADD (MODE = 1)
0 0 1	AND	1 0 1	A MINUS B (MODE = 1)
0 1 0	OR	1 1 0	MULTIPLY (F = A OR ADD)
0 1 1	SQUARE ROOT (A-B-1) (MODE = 1)	1 1 1	ABS (B) ($\neg A \wedge B$ OR $A \vee \neg B$ PLUS 1)

INSTRUCTION BITS							CONDITIONS					RESULTANT ALU SELECT SIGNALS						OPERATION
ALU 2	ALU 1	Alu 0	MODE	MULTIPLY	DIVIDE	ABS	SIGN EXT	LOW A REG LSB	HI OR LO B SIGN	LO B SIGN	ALU SIGN	S 3	S 2	S 1	S 0	CN	MODE	
0	0	0	0									L	L	L	L	H	H	$\neg$A (complement)
0	0	1	0									H	L	H	H	H	H	A $\wedge$ B (and)
0	1	0	0									H	H	H	L	H	H	A $\vee$ B (OR)
0	1	1	1									L	H	H	L	H	L	A minus B minus 1 (plus carry in)
1	0	0	1									H	L	L	H	H	L	A plus B (plus carry in)
1	0	1	1									L	H	H	L	L	L	A minus B
1	1	0	0									H	H	H	H	H	H	A
1	1	1	0									L	L	H	L	L	H	$\neg$A $\wedge$ B
				1				1					L	L			L	Used for multiply step
					1						0							FMPX bit 2= 1; $\overline{\text{ALU sign}}$ $\rightarrow$ LSB of HI B REG always
						1			1								L	A register is cleared always
							1			0		L	L	L	L			High B register is cleared always
1	1	0	0	1				1				H	L	L	H	H	L	A plus B (use for multiply)
1	1	0	1				1			0		L	L	L	L	H	L	A plus carry in } sign extend
1	1	0	1				1			1		H	H	H	H	H	L	A minus 1 plus carry in
0	0	0	1									L	L	L	L	H	L	F = A plus carry in
0	0	1	1									H	L	H	H	H	L	F = (A $\wedge$ B) minus 1 plus carry in
0	1	0	1									H	H	H	L	H	L	F = (A $\vee$ $\neg$B) plus A plus carry in
0	1	1	0									L	H	H	L	H	H	F = A exclusive or B
1	0	0	0									H	L	L	H	H	H	F = $\neg$(A exclusive or B)
1	0	1	0									L	H	H	L	H	H	F = A exclusive or B
1	1	0	1									H	H	H	H	H	L	A minus 1 plus carry in
1	1	1	1									L	L	H	L	L	L	F= (A $\vee$ $\neg$B) plus 1

Figure 12 (cont') Arithmetic control signals

Arithmetic instruction

The arithmetic unit has a 32-bit input to the A register from the A bus, a 32-bit input to the B register from the B bus, and a 32-bit output which drives the D bus. The corresponding BUS instruction mnemonics are DAREG for the A input, DBREG for the B input, and SARITH for the D bus data output. The function to be performed by the arithmetic unit is determined by executing an ALU class instruction; thereafter, the arithmetic unit will continue to perform the same function (with certain exceptions listed below) until the function is changed by executing another ALU class instruction.

The ALU instruction in the assembly language contains four specification fields which identify the arithmetic or logical function to be performed, the A input source and A register operation, the B input source and B register operation, and the F register source and its operation. The flow of data is illustrated and related to the ALU instruction bits in Figure 12.

The A and B registers are four-function bidirectional shift registers which can be controlled independently to perform arithmetic shift up, arithmetic shift down, hold, or load operations. On an ALU instruction, the A or B register will *not* be loaded from the A or B bus unless the

instruction carries the "load" code (mnemonic LD) in the field for the register to be loaded. Note also that the data loaded on an ALU instruction will not be valid unless the appropriate data buses are enabled by the assembly language .COM directive appended to the instruction.

On load or hold operations, all 32 bits are affected. For shift operations in the A register, all 32 bits are affected. For shifts in the B register, all 32 bits are affected unless the "divide" bit is set, in which case the bit shifted up from bit 15 is lost, and a quotient bit is shifted up into bit 16. In the assembly language, the "hold" mode is understood unless LD (load), SHUP (shift up), or SHDN (shift down) is written in the A or B register field.

The F register is a 32-bit clocked latch which may receive the output of the arithmetic logic element with an arithmetic shift of +1, 0, or -1 or its own output arithmetically shifted up one. The top 16 bits (16 - 31) and the bottom bits (0 - 15) are latched by independent instruction bits whose mnemonics are FENL for the low bits and FENH for the high bits. The state of the F register cannot change unless the current instruction is an ALU instruction and one or both of the F register enable bits are set.

Note in Figure 12 that the F register cannot be accessed directly from any of the data buses. Information received at the inputs of the F register and arithmetic logic element

is controlled by three four-position multiplexers. Except for one of the B inputs to the arithmetic logic element, all 32 data bits are affected similarly by the multiplexer settings. The four settings for the input to the F register have already been citied. The mnemonics are AL for the output of the arithmetic logic element (abbreviated ALU, not meaning the instruction class), ALUP for the output of the ALU shifted up 1 bit, ALDN for the output of the ALU shifted down 1 bit, and FLUP (or Ø) for the output of the F latch shifted up 1. These four states are encoded in bits 30 and 31 of the ALU class instruction.

The ALU has two inputs, designated A and B. The A input can be switched to the output of the A register (mnemonic ASA), the B register (ASB), the output of the F register (ASF), or the output of the F register shifted up 1 bit (ASFUP or Ø). These settings are controlled by bits 26 and 27 of the ALU instruction. The B input to the ALU can be the B register output (BSB), the A register output (BSA), or the F register with no shift (BSF). The fourth possible B input (mnemonic BSFUN or Ø) treats the high bits 16 through 31 differently from the low bits 0 through 15, the high 16 bits received are from the B register shifted up two bits (used for computing square roots), while the low 16 bits are received from the high 16 bits of the F register (effectively a "shift down 16 bits" command).

The latter capability is useful when splitting the 32-bit processor word into two 16-bit words to be transmitted sequentially to the host computer.

The output of the arithmetic logic element is gated onto the enabled D bus whenever the arithmetic unit is specified as source device. This output reflects the function specified by the most recent ALU class instruction operating on the F latch as then loaded and the data most recently loaded into the A and B registers. The contents of the F, A and B registers are retained, even if the processor is not running; but the ALU function is cleared to zero whenever the processor is halted.

ALU ARITHMETIC CLASS INSTRUCTION

FORMAT:

ALU Function, A-Function, B-Function, Output

This instruction performs arithmetic-logic operations and control functions on the A, B, and F registers in the ALU.

"Function" uses these operands:

ADDL add low 16 bits of A and B inputs

ADDH add high 16 bits of A and B inputs (bits 16-31)

DADD add bits 0-31 of A and B inputs

SUBL subtract bits 0-16 of A and B inputs (A minus B)

SUBH subtract bits 16-31 of A and B inputs (A minus B)

DSUB subtract bits 0-31 of A and B inputs (A minus B)

ANDH logical "and" of high 16 bits

ANDL logical "and" of low 16 bits

DAND logical "and" of all 32 bits (A and B)

ORH logical "or" of high bits

ORL logical "or" of low bits

DOR logical "or" of all 32 bits (A or B)

CLAREG clears the A register at start of instruction cycle and sets up absolute value data routing for the quantity in the B register

DPREC enables carry from bit 15 to bit 16 (redundant if 32 bit arithmetic is otherwise specified)

DIVIDE single divide step with 32 bit subtraction

A-FUNCTION:

LD - Load from A-bus

SHUP - Shift contents up 1 bit

SHDN - Shift contents down 1 bit

SA - A-input from A-register

SB - A-input from B-register

SFUP - A-input from F-register, shifted up

SF - A-input from F-register

B-FUNCTION:

LD - Load B-reg from B-bus

SHUP - Shift B-contents up 1 bit

SHDN - Shift B-contents down 1 bit

SA - B-input from A-register

SB - B-input from B-register

SF - B-input from F-register

SFUN - B-input = hi 16 bits from hi-B-latch shifted up 2
lo 16 bits from hi-F-latch

F-FUNCTION OR OUTPUT:

CLR - Clear output

LD - Load F-register

LDH - Load only upper 16 bits of F-reg

LDL - Load only lower 16 bits of F-reg

AL - F-input comes from ALU

FLUP - F-input comes from F-reg shifted up 1

ALUP
ALDN - F-input comes from ALU shifted

Branch class (jump) instructions

The fact that the next instruction to be executed is a branch class instruction is decoded during the bus cycle preceding the branch. If the branch condition is satisfied at the end of that cycle (i.e., at the beginning of the branch instruction itself), then the branch destination address is loaded into the program counter. The states of the branch conditions tested are not necessarily held over for subsequent branch tests, so the programmer must take care that an expected condition holds as of the beginning of the branch instruction of interest.

Execution of a branch entails loading the program counter with an address which may come from one of four sources: 1) bits 22-31 of the branch instruction itself, 2) the return-from-subroutine (RTS) register, 3) the D bus (which must be enabled on the *prior* instruction, or 4) the PDP-11 Unibus data lines, bits 0-9. One of these four program address sources must be specified in bits 20-21 of the branch instruction.

Bits 8 through 18 of the branch class instruction represent individual condition tests when set. If any one or more of the tested conditions is true, the branch will be executed. Otherwise the program counter will increment

as usual to fetch the next instruction in sequence. The condition tested by the "unconditional branch" bit #9 is always true.

Bit 19 of the instruction signifies a "jump to subroutine" operation. When this bit is set, the current address plus 1 is loaded into the RTS register, regardless of the results of the condition test. The branch is executed only if some tested condition is true. Thus one can program "jump to subroutine at address a if the ALU is negative" and other conditional subroutine executions.

A return from subroutine is accomplished by executing a branch instruction with the program counter address source being the RTS register. Again, some tested branch condition must be true in order for the return to occur. Note that since the RTS register can hold only one number, subroutines cannot be nested conveniently.

The branch instruction takes one bus clock cycle to execute, whether or not the branch occurs. The branch destination may be any legal program memory address.

Do-loop and repeat counters

There are three autodecrement registers which are used as counters in software loops. Two of the counters, named LOOP1 and LOOP2 in the assembly language, are tested and decremented by executing conditional branch instructions. The third counter, called the repeat counter, is used for one-instruction loops -- that is, to repeat an instruction a specified number of times.

To use a LOOP counter, the counter is ordinarily loaded with a number prior to entering the software loop by using the LLC instruction. The number loaded should be one less than the number of times the code in the loop is to be executed. A conditional branch instruction (mnemonic BRC) with the name of the LOOP counter in the test field is used to control the loop. This instruction tests the current value stored in the counter. If it is not zero, the branch is executed and the counter is decremented (by 1). If it is zero, no branch occurs,the next following instruction is executed, and the counter is not decremented.

The repeat counter contains a buffer register which indefinitely retains the repeat count loaded into it by the LRC instruction. The repeat function is activated by setting bit 2 of the instruction word. In the assembly language this is accomplished by writing .COM RE on the

line immediately following the instruction that is to be repeated. Whenever the repeat bit is set, the current contents n of the repeat count register are loaded into the repeat counter and the instruction is then executed n+1 times.

BC BRANCH CLASS INSTRUCTION

FORMAT:

BC Type, Condition, Address

This instruction permits specification of any possible branch instruction:

TYPE- uses .word ↑B0000000000111000,0 as a mask with these operands:

JSR	- Load return address	1
I	- Take branch address from instruction	Ø
DBUS	- Take branch address from D bus	2
RR	- Take branch address from return address reg	4

CONDITION - uses .word ↑B111, ↑B1111111100000000 and these operands:

LLT	- Low ALU less than Ø	4
LGE	- Low ALU not less than Ø	10
HLT	- Hi ALU negative	40
HGE	- Hi ALU positive or Ø	20
L1	- Decrement and test loop 1	1
L2	- Decrement and test loop 2	1000
CORNR	- Auto correlator not ready	2000
ALWAYS	- Unconditional	2

ADDRESS - uses .word ↑B1111111111000000,0 **Any** predefined or user defined symbol or valid number may appear here, not including special symbols.

BRANCH ON CONDITION

INSTRUCTION FORMAT:

BRC Test, addr

Where addr is any valid address in the program memory. The first operand, test is a mask specifying the branch conditions. The test may be specified by one (or the sum of more than one) of the following special symbols:

LLT - Low ALU is less than zero

LGE - Low Alu is not less than zero

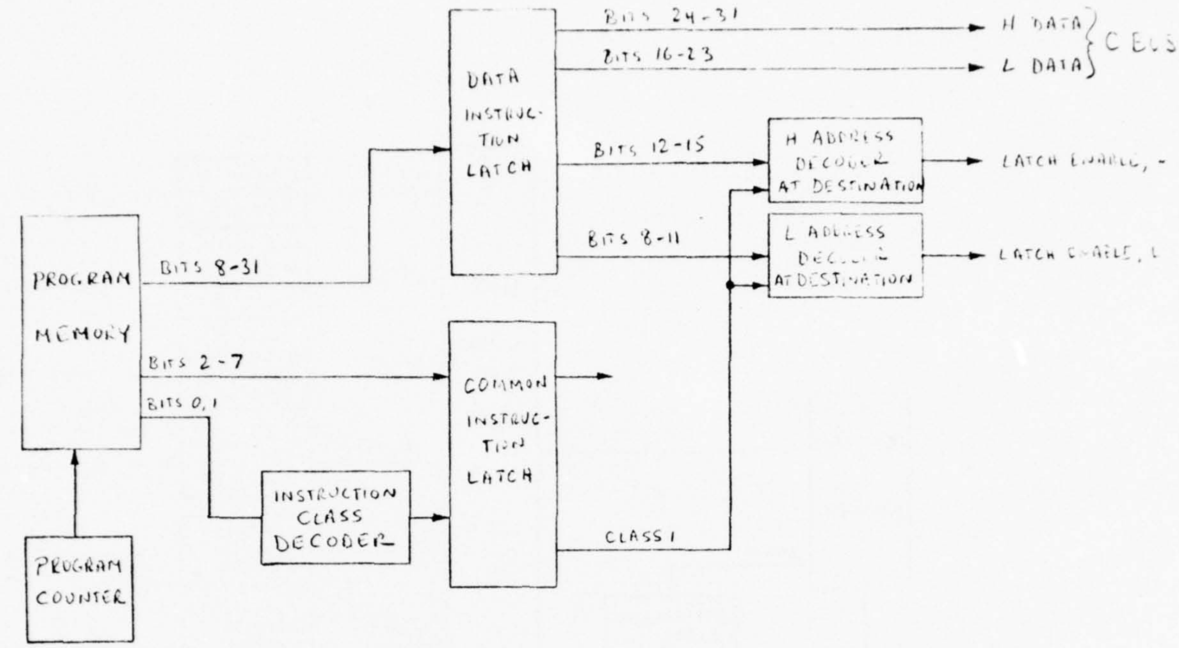
HLT - High Alu is less than zero

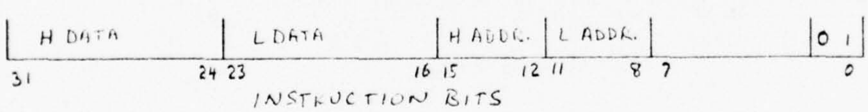
HGE - High ALU is not less than zero

CORNR- Auto correlator not ready

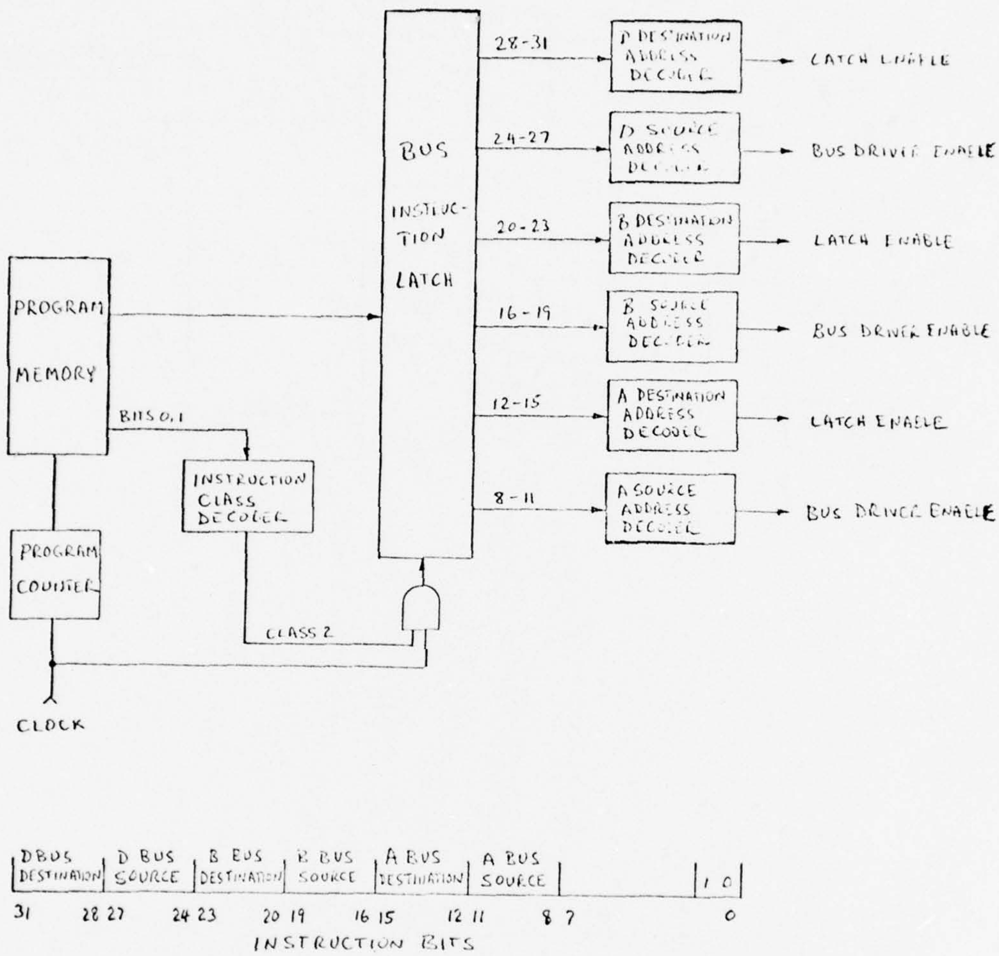
L1-
L2- { } decrement loop counter and branch if zero

Additional information on the architecture of the vector processor is contained in the accompanying diagrams.

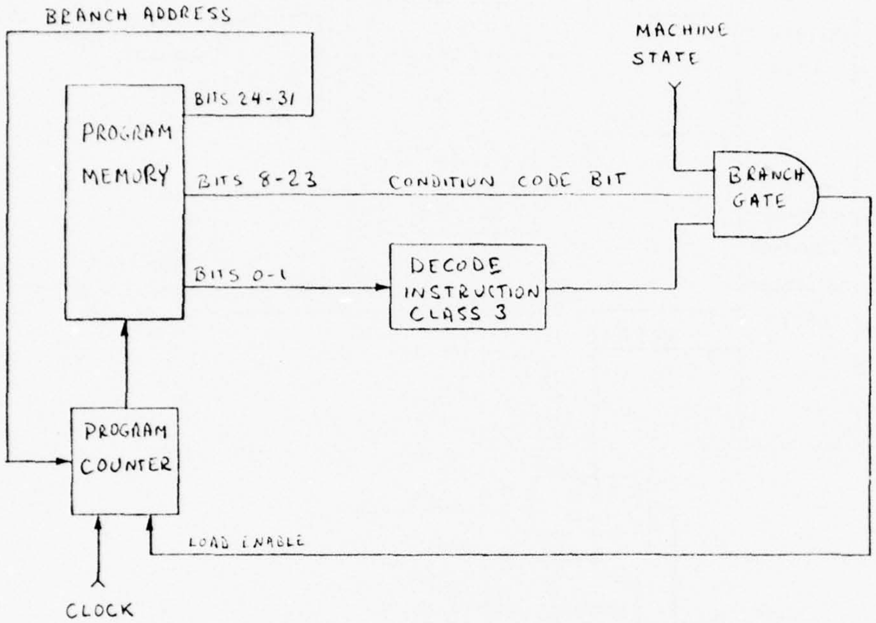


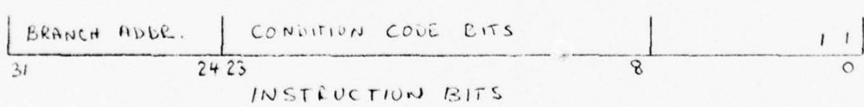


DATA INSTRUCTION

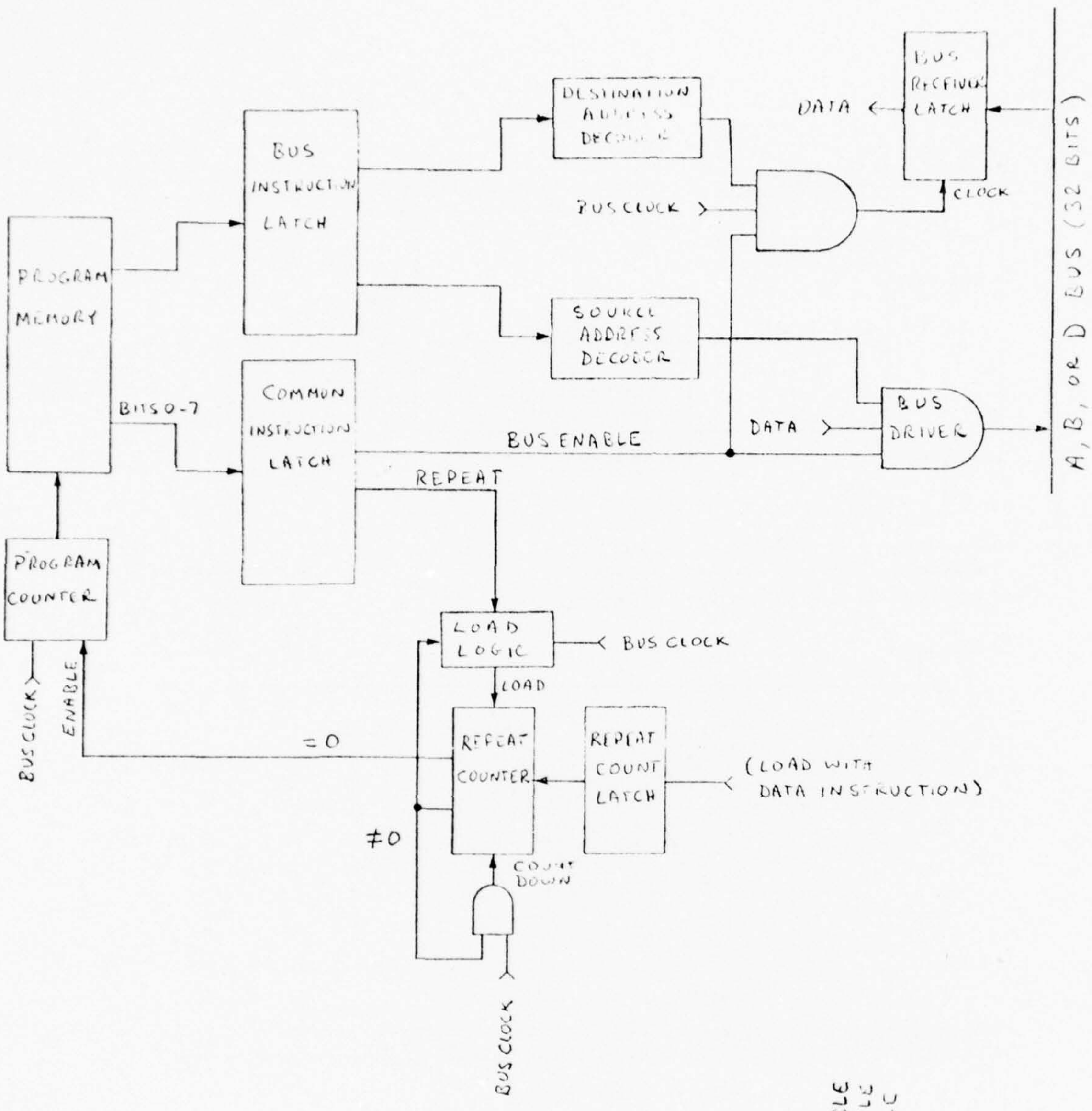


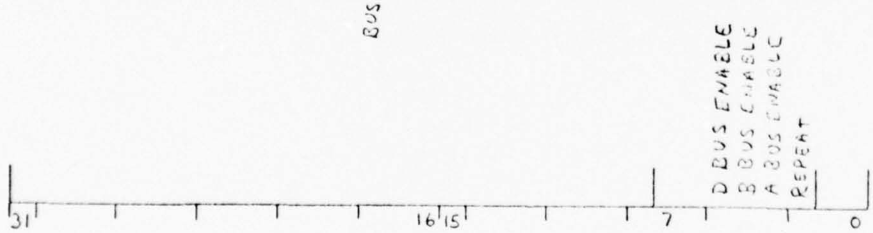
BUS INSTRUCTION



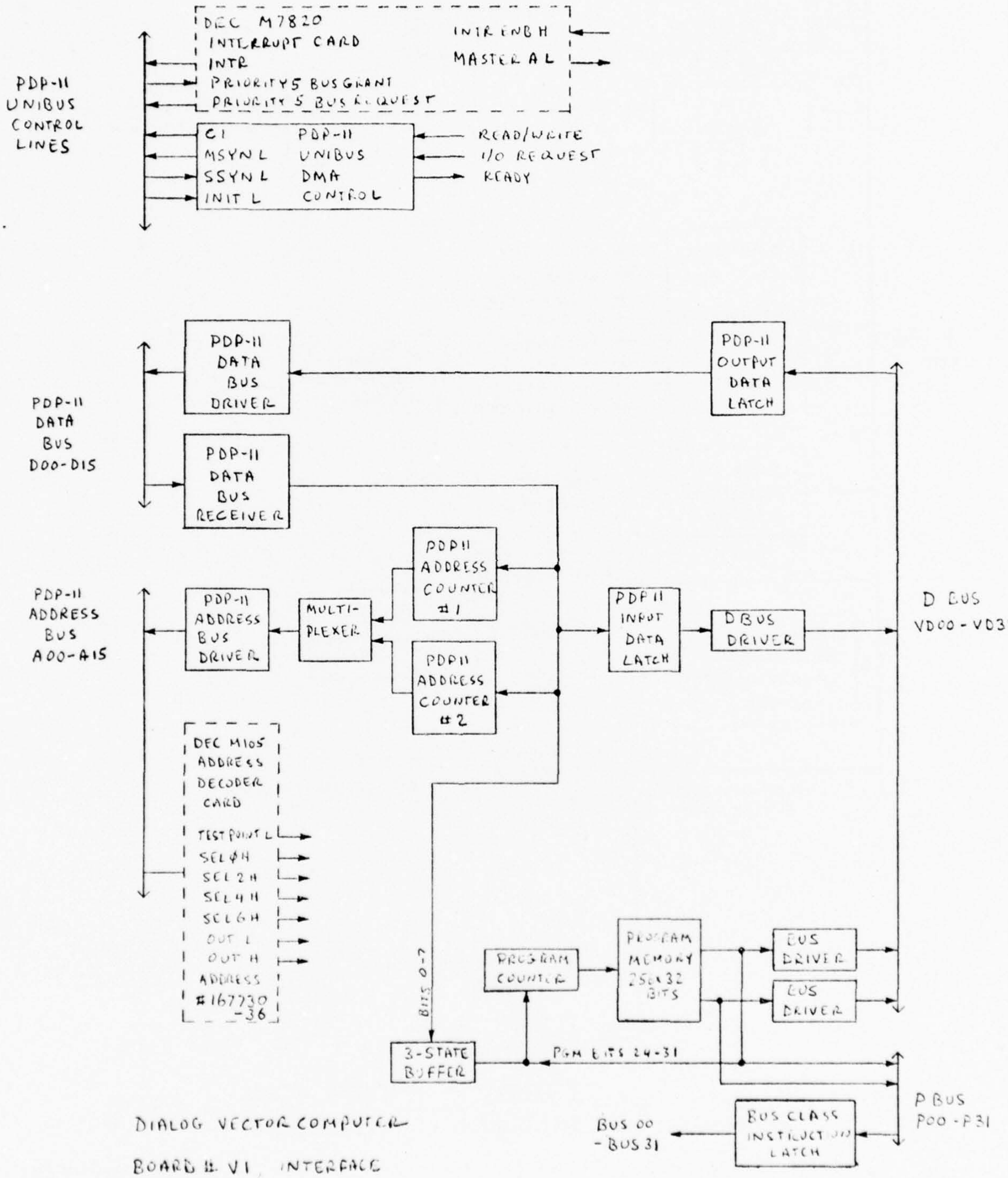


BRANCH INSTRUCTION





COMMON INSTRUCTION BITS

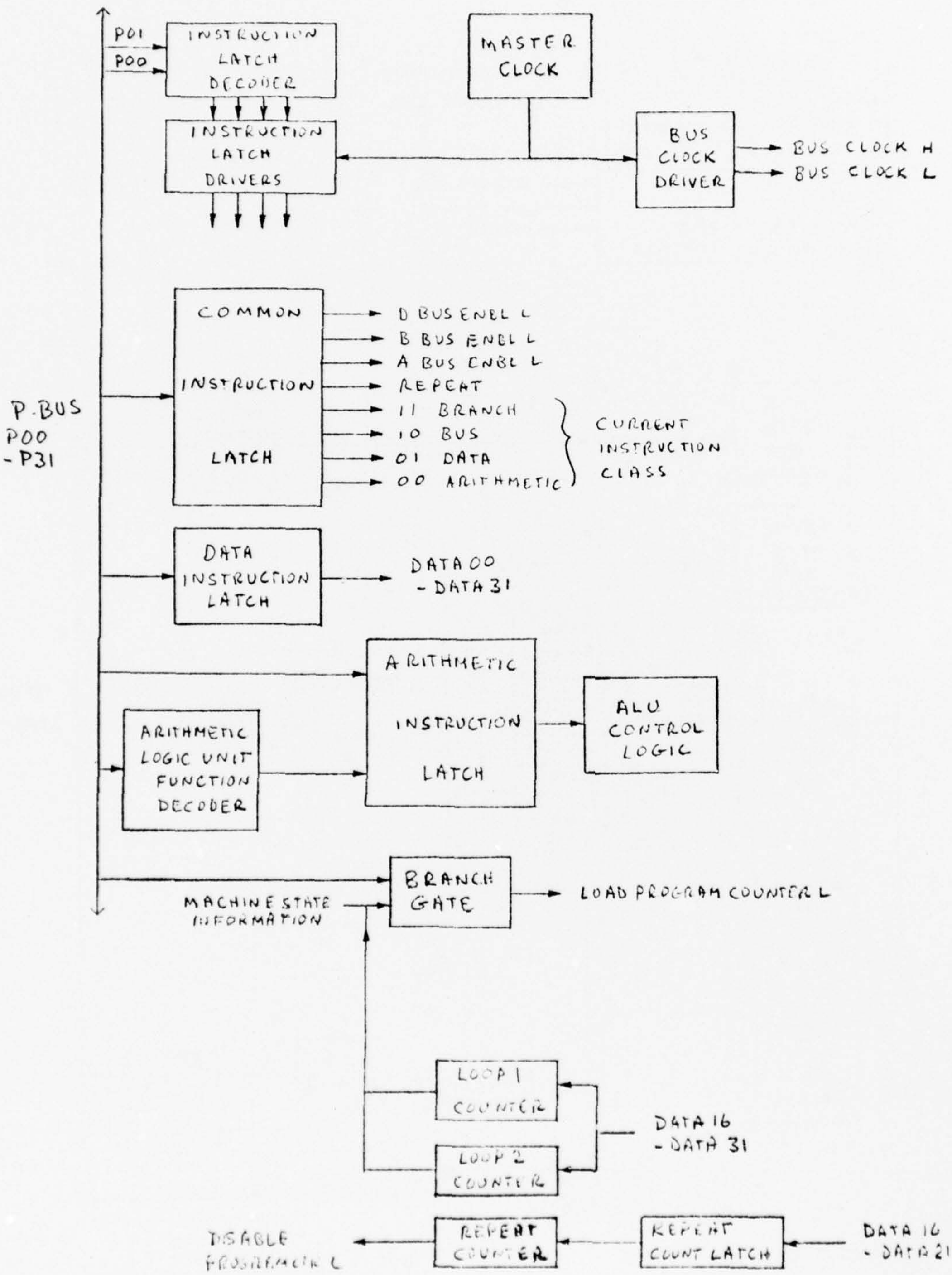


DIALOG VECTOR COMPUTER

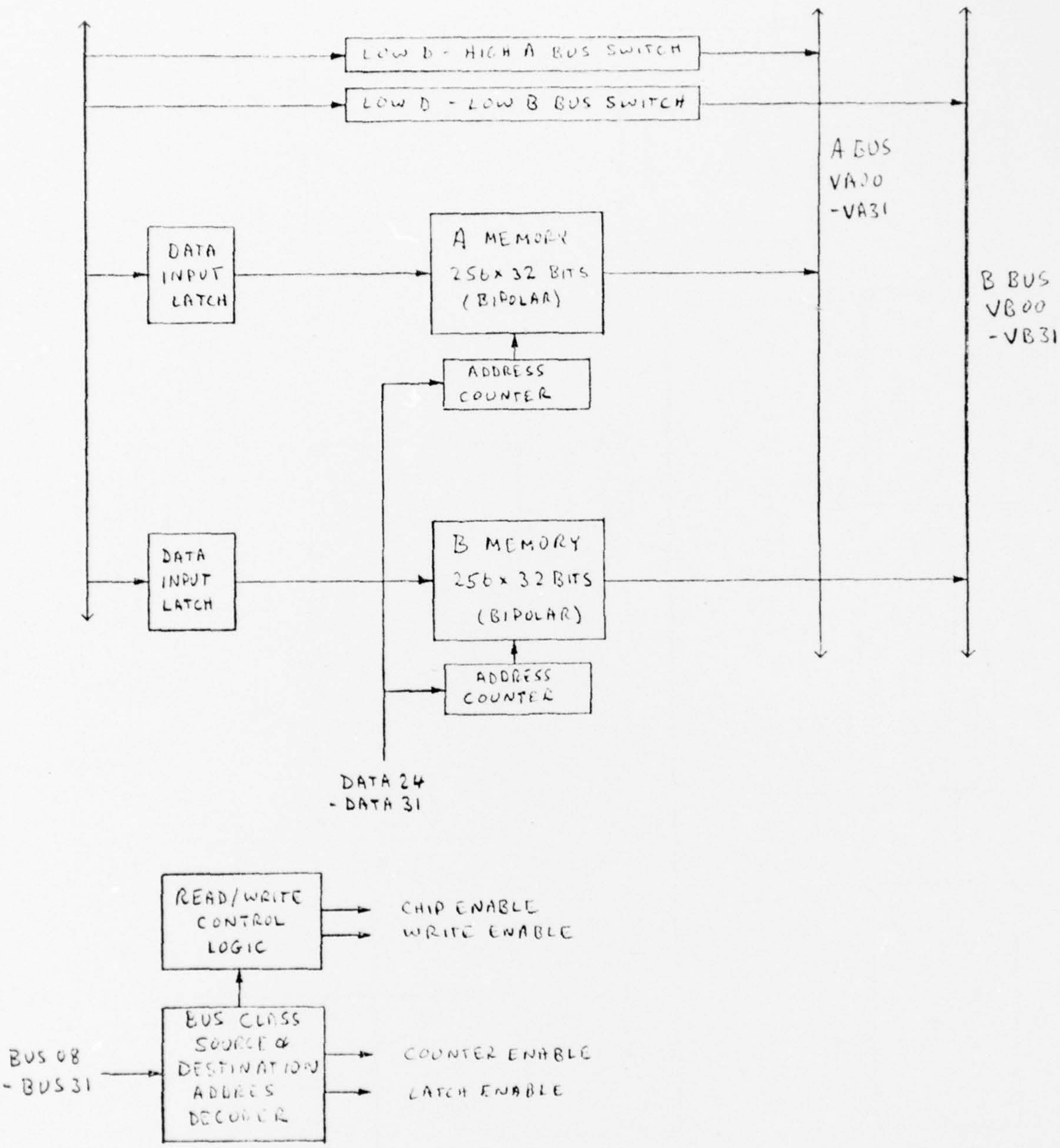
BOARD # V1 INTERFACE

BLOCK DIAGRAM

13 AUG 75



DIALOG VECTOR COMPUTER
BOARD II V2 - INSTRUCTION DECODER
BLOCK DIAGRAM
15 AUG 75

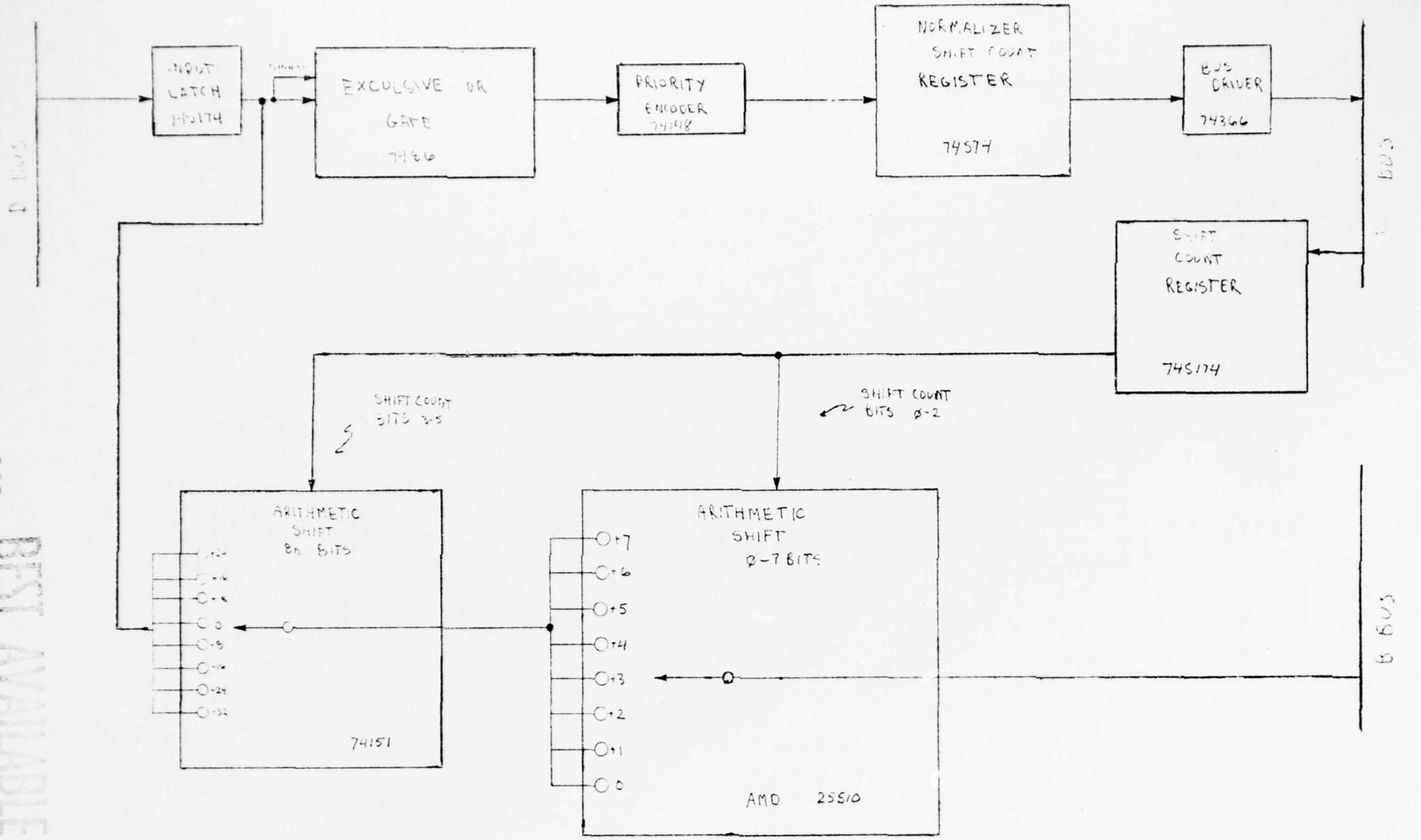


DIALOG VECTOR COMPUTER

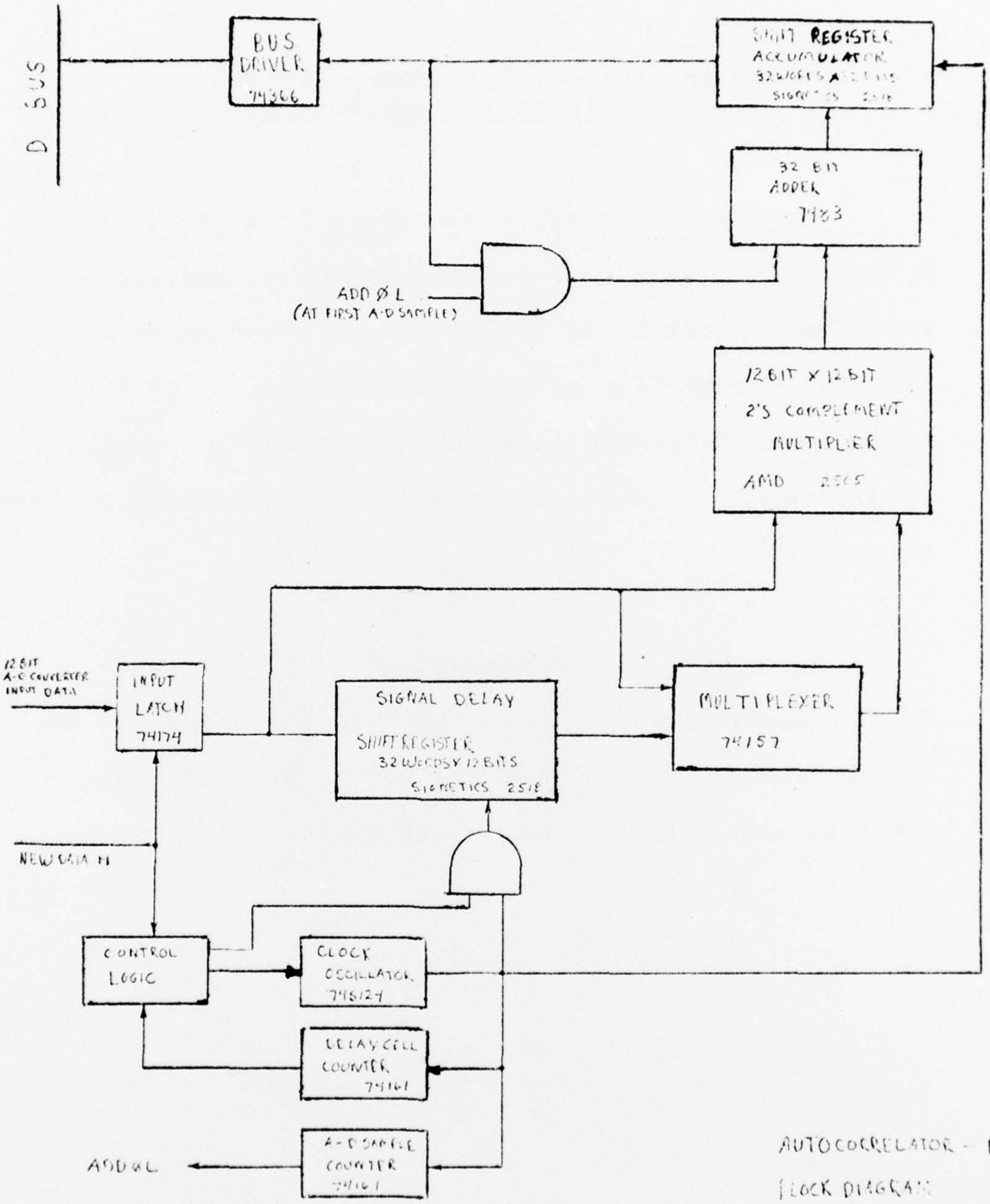
BOARD # 13 - MEMORY

BLOCK DIAGRAM

13 AUG 75



SHIFTER-BOARD #V6
BLOCK DIAGRAM
19-Sept-75



AUTOCORRELATOR - BOARD # V7
BLOCK DIAGRAM
19 Sept 75

IV Appendix

Tables of Lower Confidence Limits for the Binomial Distribution.

Suppose there are K successes in a sequence of N Bernoulli trials. For these variables and a chosen confidence level C the tables give a lower bound P on the true probability of success per trial. If the true probability were lower than P, then the probability of getting K or more successes in N trials would be less than 1-C.

Example: An experiment yields 49 successes out of 50 independent trials. From the table for N = 50, K = 49, C = 0.9, find that the 90% lower confidence limit on the true probability is P = 0.9244.

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 5

K	C= K/N	0.99	0.98	0.95	0.90
5	1.0000	0.3981	0.4573	0.5493	0.6309
4	0.8000	0.2221	0.2671	0.3426	0.4161
3	0.6000	0.1056	0.1353	0.1892	0.2466

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 10

K	C= K/N	0.99	0.98	0.95	0.90
10	1.0000	0.6309	0.6763	0.7411	0.7943
9	0.9000	0.4956	0.5398	0.6058	0.6632
8	0.8000	0.3883	0.4295	0.4931	0.5504
7	0.7000	0.2971	0.3343	0.3934	0.4483
6	0.6000	0.2183	0.2507	0.3035	0.3542
5	0.5000	0.1504	0.1773	0.2224	0.2673

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 15

K	C= K/N	0.99	0.98	0.95	0.90
15	1.0000	0.7357	0.7704	0.8190	0.8577
14	0.9333	0.6321	0.6683	0.7206	0.7644
13	0.8667	0.5468	0.5830	0.6366	0.6827
12	0.8000	0.4715	0.5069	0.5602	0.6072
11	0.7333	0.4031	0.4371	0.4892	0.5360
10	0.6667	0.3403	0.3725	0.4226	0.4683
9	0.6000	0.2823	0.3123	0.3596	0.4035
8	0.5333	0.2287	0.2561	0.3000	0.3415

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 20

K	C= K/N	0.99	0.98	0.95	0.90
20	1.0000	0.7943	0.8224	0.8609	0.8913
19	0.9500	0.7112	0.7412	0.7839	0.8190
18	0.9000	0.6417	0.6725	0.7174	0.7552
17	0.8500	0.5793	0.6104	0.6563	0.6958
16	0.8000	0.5217	0.5527	0.5990	0.6394
15	0.7500	0.4679	0.4984	0.5444	0.5851
14	0.7000	0.4171	0.4469	0.4922	0.5327
13	0.6500	0.3691	0.3978	0.4420	0.4820
12	0.6000	0.3234	0.3509	0.3936	0.4327
11	0.5500	0.2801	0.3061	0.3469	0.3848
10	0.5000	0.2390	0.2633	0.3019	0.3382

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 25

K	C= K/N	0.99	0.98	0.95	0.90
25	1.0000	0.8318	0.8551	0.8871	0.9120
24	0.9600	0.7625	0.7880	0.8239	0.8531
23	0.9200	0.7041	0.7307	0.7690	0.8009
22	0.8800	0.6512	0.6786	0.7183	0.7520
21	0.8400	0.6021	0.6298	0.6704	0.7053
20	0.8000	0.5557	0.5835	0.6246	0.6603
19	0.7600	0.5116	0.5392	0.5805	0.6167
18	0.7200	0.4694	0.4967	0.5378	0.5742
17	0.6800	0.4289	0.4557	0.4964	0.5327
16	0.6400	0.3900	0.4161	0.4561	0.4921
15	0.6000	0.3524	0.3778	0.4168	0.4523
14	0.5600	0.3163	0.3407	0.3786	0.4133
13	0.5200	0.2814	0.3048	0.3414	0.3752

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 30

K	C= K/N	0.99	0.98	0.95	0.90
30	1.0000	0.8577	0.8777	0.9050	0.9261
29	0.9667	0.7984	0.8205	0.8514	0.8764
28	0.9333	0.7481	0.7715	0.8047	0.8322
27	0.9000	0.7024	0.7266	0.7614	0.7907
26	0.8667	0.6597	0.6844	0.7204	0.7510
25	0.8333	0.6192	0.6443	0.6810	0.7126
24	0.8000	0.5805	0.6057	0.6430	0.6753
23	0.7667	0.5433	0.5685	0.6061	0.6388
22	0.7333	0.5073	0.5325	0.5701	0.6032
21	0.7000	0.4726	0.4974	0.5349	0.5681
20	0.6667	0.4388	0.4634	0.5006	0.5337
19	0.6333	0.4060	0.4302	0.4669	0.4999
18	0.6000	0.3742	0.3978	0.4339	0.4666
17	0.5667	0.3432	0.3662	0.4016	0.4338
16	0.5333	0.3132	0.3354	0.3699	0.4015
15	0.5000	0.2839	0.3054	0.3389	0.3697

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 35

K	C= K/N	0.99	0.98	0.95	0.90
35	1.0000	0.8767	0.8943	0.9180	0.9363
34	0.9714	0.8249	0.8444	0.8715	0.8934
33	0.9429	0.7808	0.8015	0.8308	0.8550
32	0.9143	0.7406	0.7622	0.7931	0.8190
31	0.8857	0.7028	0.7251	0.7573	0.7845
30	0.8571	0.6670	0.6897	0.7228	0.7510
29	0.8286	0.6326	0.6557	0.6894	0.7185
28	0.8000	0.5995	0.6227	0.6569	0.6866
27	0.7714	0.5673	0.5907	0.6252	0.6554
26	0.7429	0.5361	0.5594	0.5942	0.6246
25	0.7143	0.5058	0.5290	0.5637	0.5944
24	0.6857	0.4761	0.4992	0.5338	0.5646
23	0.6571	0.4472	0.4700	0.5045	0.5352
22	0.6286	0.4189	0.4414	0.4756	0.5062
21	0.6000	0.3913	0.4134	0.4472	0.4775
20	0.5714	0.3643	0.3860	0.4192	0.4492
19	0.5429	0.3379	0.3591	0.3917	0.4213
18	0.5143	0.3120	0.3327	0.3646	0.3937

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 40

K	C= K/N	0.99	0.98	0.95	0.90
40	1.0000	0.8913	0.9068	0.9278	0.9440
39	0.9750	0.8457	0.8627	0.8868	0.9062
38	0.9500	0.8060	0.8246	0.8509	0.8724
37	0.9250	0.7701	0.7896	0.8174	0.8405
36	0.9000	0.7364	0.7566	0.7856	0.8100
35	0.8750	0.7043	0.7250	0.7550	0.7804
34	0.8500	0.6734	0.6945	0.7252	0.7515
33	0.8250	0.6435	0.6649	0.6963	0.7233
32	0.8000	0.6146	0.6362	0.6680	0.6955
31	0.7750	0.5863	0.6081	0.6402	0.6682
30	0.7500	0.5588	0.5806	0.6129	0.6412
29	0.7250	0.5319	0.5536	0.5861	0.6146
28	0.7000	0.5055	0.5272	0.5597	0.5884
27	0.6750	0.4797	0.5013	0.5337	0.5625
26	0.6500	0.4542	0.4758	0.5081	0.5368
25	0.6250	0.4295	0.4507	0.4828	0.5114
24	0.6000	0.4051	0.4260	0.4578	0.4863
23	0.5750	0.3812	0.4017	0.4331	0.4614
22	0.5500	0.3577	0.3779	0.4088	0.4368
21	0.5250	0.3346	0.3544	0.3848	0.4124
20	0.5000	0.3119	0.3312	0.3611	0.3883

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 50

K	C= K/N	0.99	0.98	0.95	0.90
50	1.0000	0.9120	0.9247	0.9418	0.9550
49	0.9800	0.8745	0.8888	0.9086	0.9244
48	0.9600	0.8423	0.8577	0.8794	0.8970
47	0.9400	0.8128	0.8291	0.8522	0.8713
46	0.9200	0.7850	0.8020	0.8262	0.8465
45	0.9000	0.7584	0.7760	0.8012	0.8224
44	0.8800	0.7328	0.7508	0.7768	0.7989
43	0.8600	0.7080	0.7264	0.7531	0.7758
42	0.8400	0.6839	0.7025	0.7298	0.7531
41	0.8200	0.6603	0.6792	0.7069	0.7308
40	0.8000	0.6372	0.6563	0.6844	0.7087
39	0.7800	0.6146	0.6338	0.6622	0.6869
38	0.7600	0.5923	0.6117	0.6403	0.6653
37	0.7400	0.5705	0.5899	0.6187	0.6440
36	0.7200	0.5490	0.5684	0.5974	0.6228
35	0.7000	0.5278	0.5472	0.5763	0.6018
34	0.6800	0.5070	0.5263	0.5554	0.5810
33	0.6600	0.4864	0.5057	0.5347	0.5604
32	0.6400	0.4661	0.4853	0.5142	0.5399
31	0.6200	0.4461	0.4652	0.4940	0.5196
30	0.6000	0.4263	0.4453	0.4739	0.4995
29	0.5800	0.4069	0.4256	0.4540	0.4795
28	0.5600	0.3876	0.4061	0.4343	0.4596
27	0.5400	0.3686	0.3869	0.4147	0.4399
26	0.5200	0.3499	0.3679	0.3954	0.4203
25	0.5000	0.3314	0.3491	0.3763	0.4009

CONFIDENCE LEVELS FOR BINOMIAL DISTRIBUTION
P SUCH THAT PROB[X<K GIVEN P] = C

N= 60

K	C= K/N	0.99	0.98	0.95	0.90
60	1.0000	0.9261	0.9369	0.9513	0.9624
59	0.9833	0.8944	0.9066	0.9234	0.9367
58	0.9667	0.8672	0.8803	0.8988	0.9137
57	0.9500	0.8422	0.8561	0.8758	0.8920
56	0.9333	0.8185	0.8331	0.8539	0.8712
55	0.9167	0.7959	0.8111	0.8327	0.8509
54	0.9000	0.7741	0.7897	0.8122	0.8311
53	0.8833	0.7529	0.7689	0.7920	0.8116
52	0.8667	0.7322	0.7485	0.7723	0.7924
51	0.8500	0.7120	0.7286	0.7529	0.7735
50	0.8333	0.6922	0.7091	0.7337	0.7549
49	0.8167	0.6727	0.6898	0.7148	0.7364
48	0.8000	0.6536	0.6708	0.6962	0.7181
47	0.7833	0.6347	0.6521	0.6778	0.7000
46	0.7667	0.6161	0.6336	0.6595	0.6820
45	0.7500	0.5978	0.6154	0.6414	0.6642
44	0.7333	0.5797	0.5973	0.6236	0.6465
43	0.7167	0.5618	0.5795	0.6058	0.6289
42	0.7000	0.5441	0.5618	0.5883	0.6115
41	0.6833	0.5266	0.5444	0.5708	0.5942
40	0.6667	0.5093	0.5271	0.5535	0.5769
39	0.6500	0.4923	0.5099	0.5364	0.5598
38	0.6333	0.4753	0.4929	0.5193	0.5428
37	0.6167	0.4586	0.4761	0.5024	0.5259
36	0.6000	0.4420	0.4594	0.4857	0.5090
35	0.5833	0.4256	0.4429	0.4690	0.4923
34	0.5667	0.4093	0.4265	0.4524	0.4757
33	0.5500	0.3933	0.4102	0.4360	0.4591
32	0.5333	0.3773	0.3941	0.4197	0.4427
31	0.5167	0.3616	0.3782	0.4035	0.4263
30	0.5000	0.3460	0.3624	0.3874	0.4101